Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В.Непомнящий
подпись инициалы, фамилия
« ___» _____ 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Моделирование и прототипирование основных узлов для встроенного сетевого контроллера защищенного канала Ethernet
тема

Руководитель	_____ подпись, дата	профессор, зав. Каф. ВТ, к.т.н. должность, ученая степень	О.В.Непомнящий инициалы, фамилия
Выпускник	_____ подпись, дата		С.Л.Лещенко инициалы, фамилия
Нормоконтролер	_____ подпись, дата	доцент, к.т.н. должность, ученая степень	В.И. Иванов инициалы, фамилия

Красноярск 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Проектирование сетевых контроллеров из состава аппаратуры специального назначения, предназначенной для эксплуатации в условиях дестабилизирующих факторов окружающей среды, является актуальной инженерной задачей. Основными проблемами проектирования являются обеспечение надежности связи, защита от несанкционированного доступа, компактная реализация, радиационная стойкость устройств, а также высокие скорости передачи информации [1].

Для решения обозначенных задач разрабатываются нестандартные протоколы и методики обмена данными, применяется помехоустойчивое кодирование и шифрование. На аппаратном уровне обеспечивают радиационную стойкость. Поэтому создание доступных инструментальных средств, обеспечивающих отработку новых протоколов высокого уровня, является актуальной инженерной задачей.

В составе бортовой аппаратуры зачастую применяются устройства, подключенные к Ethernet каналу. Например: навигационные приемники наземного базирования, контрольно-измерительная аппаратура, станции тропосферной связи и прочее [2].

В отечественной промышленности идет активное импортозамещение, что предполагает использование отечественных аналогов, которых не всегда достаточно для производства той или иной составляющей. В России выпускается ряд готовых решений на базе радиационно-стойких сверхбольших интегральных схем (СБИС). Например, СБИС от компаний Элвис [3] или Миландр [4]. Однако в данных СБИС противодействие дестабилизирующим факторам реализуется при разработке на техническом уровне. Например, применяют технологии «кремний на изоляторе», «кремний на сапфире», охранные кольца для ячеек, дублирование или троирование ячеек памяти и прочее [5].

Таким образом, целью выпускной квалификационной работы является разработка программно-аппаратного инструментария для отработки программного обеспечения сетевых контроллеров стандарта Ethernet.

1 Обзор ведущих мировых достижений в реализации защищенных каналов связи стандарта Ethernet

1.1 Обзор сетевых устройств

Известен ряд компаний, выпускающих контроллеры со встроенным Ethernet. Например: Microchip, Intel, WIZnet, ЭЛВИС, Миландр [3-4, 6-8].

1.1.1 Microchip

Занимается разработкой микроконтроллеров, интерфейсных (CAN, Ethernet, IrDA, LIN, I2C, SPI) и аналоговых микросхем.

Для реализации простейших сетей можно выделить несколько контроллеров из семейства PIC18 со встроенным Ethernet серии J. Используемый объем памяти варьируется от 32 Кб до 128 Кб, поддерживаются протоколы связи USB, TCP/IP и работают в диапазоне напряжений 2.0...3.6 В [6].

Рассмотрим один из Ethernet-контроллеров, ENC28J60. В нем используются интерфейсы SPI, I2C и поддерживаются протоколы 10BASE-T/100BASE-TX/1000BASE-FX 802.3. Реализуется прием/передача пакетных данных с контролем ошибок. Вся память (128 Кб) делится на буферную, управляющие регистры и регистры физического уровня. Рабочие температуры в диапазоне от -40°C до +85°C. Потребляемые параметры 250 мА и 3.3/5 В.

Кроме того, на базе микроконтроллера ENC28J60 построен Ethernet Shield, предназначенный для удобства его программирования. Он включает в себя протокол приема и передачи данных, MAC адрес и протокол PHY. Используется интерфейс RJ-45 MAG-Jack со встроенным трансформатором [9]. Встроен стабилизатор питания на 3,3 В [6].

1.1.2 Intel

Intel является ведущим производителем микропроцессоров и микроконтроллеров. Известно более 30 сетевых контроллеров, отвечающих требованиям ВКР [7].

Рассмотрим основные характеристики на базе Ethernet-контроллера X540-AT2. В нем есть поддержка протоколов 10/100BASE-T IEEE 802.3 (Ethernet), IEEE 1588 (PTP), IEEE 802.1 AE (MAC Security) и системный интерфейс PCIe v2.1.

В других контроллерах, в зависимости от года выпуска, меняется только скорость передачи пакетов, рабочие температуры устройства (от -40°C до +85°C), тип системного интерфейса PCI (от v1.0 до v3.0), а также наличие дополнительных интерфейсов (SFI, KR, KR4, XAUI, KX, KX4, SGMII, BX, BX4, CX4, 10/100/1000BASE-T, SERDES).

1.1.3 WIZnet

Известен ряд устройств, разработанных данной фирмой. К примеру, Ethernet Shield W5500 предназначен для приема и передачи пакетов на плате Arduino/ARM с чипом W5500. Данный модуль поддерживает напряжение в 3.3 и 5 В. Есть такие интерфейсы как RJ-45, I2C, UART, SPI и поддержка протокола 10BASE-T IEEE 802.3 [8].

1.1.4 ЭЛВИС

Наибольший интерес вызывают отечественные разработки. ЭЛВИС в основном выпускает радиационно-стойкие микросхемы для космических аппаратов и широкополосных систем связи, а также процессоры "Мультикор".

Имеет достаточно обширный выбор микропроцессоров и контроллеров на базе сетей SpaceWire/GigaSpaceWire.

Например, микросхема 1892ВМ14Я (рисунок 1) предназначена для систем управления и обработки данных в современных сетях и включает в себя передачу пакетной информации. Разработана на базе радиационно-стойких библиотек МК180RT разработки АО НПЦ «ЭЛВИС» и IP-библиотек платформы «МУЛЬТИКОР». Поддерживает напряжение в 3.3В. В ней расположено 2 порта SpaceWire со скоростью передачи данных от 2 до 300 Мбит/с, 2 порта UART, Ethernet MAC 10/100/1000 Мбит/с несколько таймеров (реального времени и интервальный). Работает только в операционных системах Linux и ОС реального времени uOS [3].

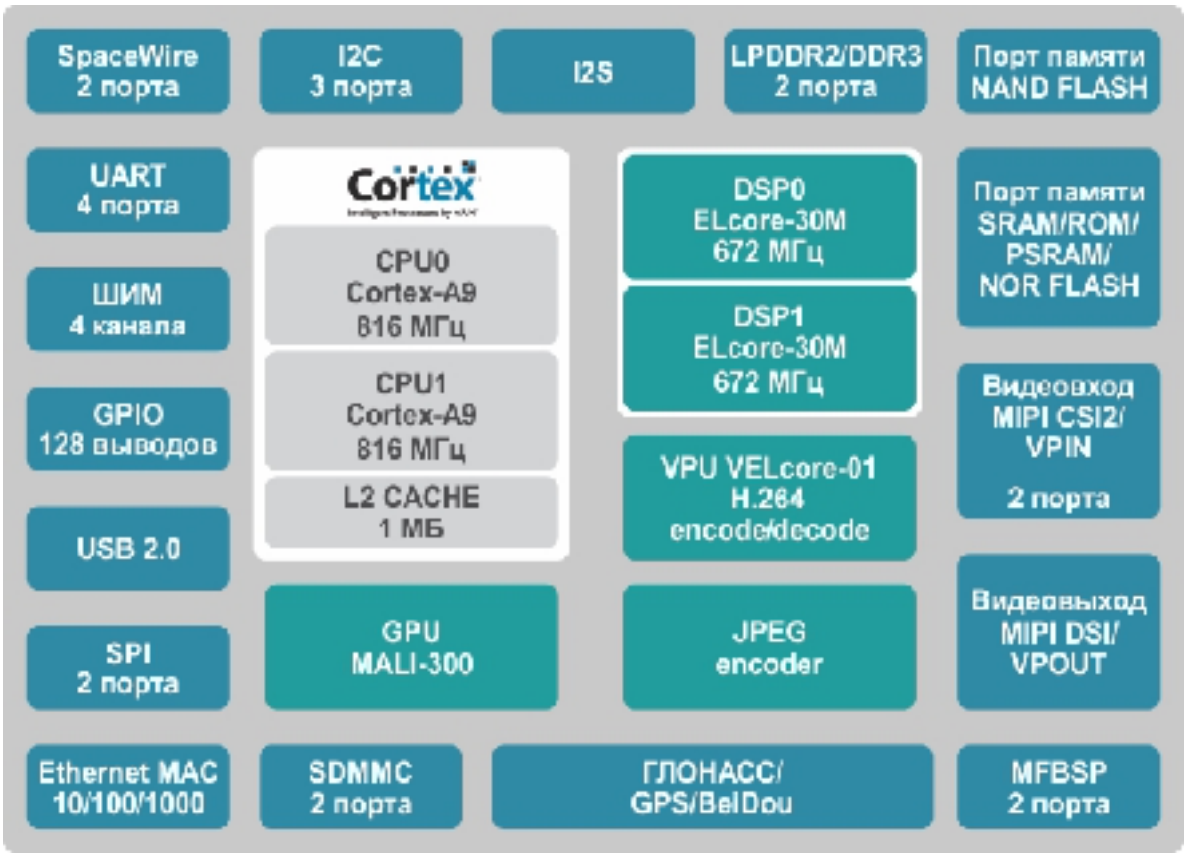


Рисунок 1 – Структурная схема 1892ВМ14Я

1.1.5 Миландр

Данная фирма производит интегральные микросхемы, электронные модули, различные приборы и системы. Имеет достаточно обширный выбор микропроцессоров и контроллеров на базе интерфейсов CAN, Ethernet, LIN, RS-232/485/422.

Например, микросхема контроллера 5600ВГ1У (рисунок 2) предназначена для работы в ЛВС сетях на основе протокола 10BASE-T IEEE 802.3. Реализует прием/передачу пакетных данных с контролем ошибок. Поддерживает напряжение в 3.3 В. Температурный диапазон от -60°С до +86°С [4].

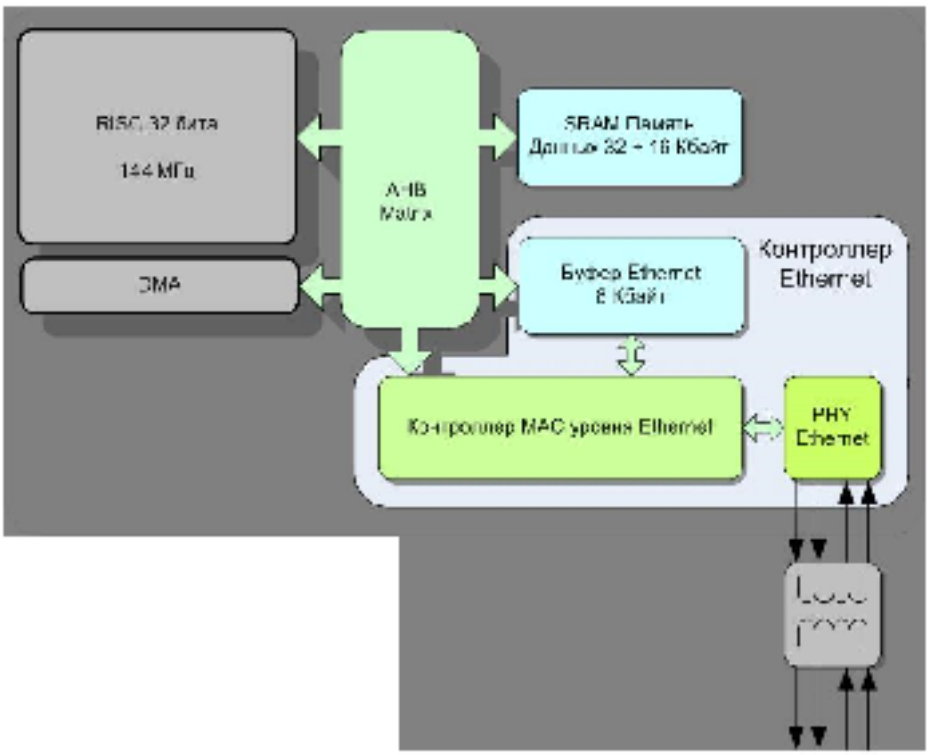


Рисунок 2 – Структурная схема 5600ВГ1У

В таблице 1 приведено сравнение основных характеристик устройств производителей: Microchip - Ethernet Shield ENC28J60; Intel - X540-AT2; WIZnet -

Ethernet Shield W5500; ЭЛВИС - 1892ВМ14Я; Миландр - 5600ВГ1У. Выбор микросхем произведен так, чтобы представить каждую фирму среднестатистическими моделями, которые имеют развитую структуру.

На основе результатов анализа выбран Ethernet контроллер для реализации. В Ethernet Shield на базе микроконтроллера ENC28J60 используется интерфейс RJ-45 MAG-Jack со встроенным трансформатором. Также встроен стабилизатор питания на 3,3 В, но на входах интерфейса SPI используется до 6 В. Потребляемый ток 250 мА, а частота работы 25 МГц.

Таблица 1 – Сравнительная таблица основных характеристик сетевых устройств

Фирма	Microchip	Intel	WIZnet	ЭЛВИС	Миландр
Устройство	Ethernet Shield ENC28J60	X540-AT2	Ethernet Shield W5500	1892ВМ14Я	5600ВГ1У
Напряжение питания, В	3.3, 5	5	3.3, 5	3,3	3,3
Потребляемый ток, мА	250	300	250	250	250
Рабочие температуры, °С	-40 – +85	-40 – +85	-40 – +85	-40 – +85	-60 – +85
Поддерживаемые протоколы	10BASE-T/100BASE-TX/1000BASE-FX IEEE 802.3, IEEE 1588, IEEE 802.1 AE	10BASE-T/100BASE-TX IEEE 802.3, IEEE 1588, IEEE 802.1 AE	10BASE-T IEEE 802.3	SpaceWire от 2 до 300Мбит/с, 10BASE-T/100BASE-TX/1000BASE-FX IEEE 802.3	10BASE-T/100BASE-TX/1000BASE-FX IEEE 802.3, IEEE 1588, IEEE 802.1 AE
Объем памяти, Кбайт	128	64	32 + 16	128	8 + 32 + 32 +16
Имеющиеся интерфейсы	SPI, I2C, RJ-45 MAG-Jack	PCIe, I2C	RJ-45, I2C, UART, SPI	SpaceWire 2 порта, UART 2 порта, Ethernet MAC	RS-232, I2C
Встроенный таймер	+	+	-	+	-
Поддержка Ethernet	+	+	+	-	+
Радиационная стойкость	+	+	+	+	+

1.2 Обзор микроконтроллеров

Микроконтроллеры фирмы Atmel широко применяются в компьютерных сетях, космосе, военных устройствах, медицине, промышленности и т.д. Продукция этой фирмы включает в себя ядра собственной разработки. Также Atmel разрабатывает собственное семейство микроконтроллеров AVR. Данное семейство имеет гарвардскую архитектуру и расширенную систему команд RISC. Процессор AVR имеет 32 8-битных регистра общего назначения, объединенных в регистровый файл [6].

Рассмотрим несколько контроллеров фирмы Atmel в сравнительной таблице 2 и выявим наиболее подходящий, по характеристикам, для разработки системы.

Таблица 2 – Сравнительная таблица микроконтроллеров фирмы Atmel

Микроконтроллер	ATmega8	ATmega48	ATmega88	ATmega168	ATmega328	ATmega8535
FLASH, КБайт	8	4	8	16	32	8
EEPROM, байт	512	256	512	512	1024	512
SRAM, байт	1024	512	1024	1024	2048	512
Внешнее ОЗУ	-	-	-	-	-	-
Кол-во команд	130	131	131	131	131	130
Кол-во портов I/O	23	23	23	23	23	32
Кол-во внешних источников прерываний	2	24	24	24	24	3
Таймеры	2 - 8разр., 1 - 16разр.	2 - 8разр., 1 - 16разр.	2 - 8разр., 1 - 16разр.	2 - 8разр., 1 - 16разр.	2 - 8разр., 1 - 16разр.	2 - 8разр., 1 - 16разр.
ШИМ	3	6	6	6	6	4
SPI	1	2	2	2	2	1
I2C	1	1	1	1	1	1
UART	1	1	1	1	1	1
АЦП (10 разр.)	6	6	6	6	8	8
Рабочая частота, МГц	0 ... 16	0 ... 10 0 ... 20	0 ... 10 0 ... 20	0 ... 10 0 ... 20	0 ... 4 0 ... 10 0 ... 20	0 ... 16
Напряжение питания, В	4.5 ... 5.5	2.7 ... 5.5 4.5 ... 5.5	2.7 ... 5.5 4.5 ... 5.5	2.7 ... 5.5 4.5 ... 5.5	1.8 ... 5.5 2.7 ... 5.5 4.5 ... 5.5	4.5 ... 5.5
Кол-во DIP	28	28	28	28	28	40
Кол-во TQFP	32	32	32	32	32	44
Кол-во MLF	32	28	32	32	28	44

В результате сравнения микроконтроллеров стоит заметить, что характеристики устройств подобны. Из-за того, что необходим большой объем памяти для приема и передачи пакетов, лучшим вариантом является ATmega328. Память под программу составляет 32 Кбайт FLASH памяти, а под данные: EEPROM память 1024 байт и SRAM 2048 байт. Используется 10 разрядный АЦП.

Данный микроконтроллер располагается на программаторе Arduino Nano. Для подключения Ethernet модуля к программатору используется интерфейс SPI. Используемое напряжение питания составляет 5 В, а рабочая частота, в зависимости от скорости передачи пакетов варьируется от 0 до 20 МГц.

1.3 Обзор средств моделирования

Для проверки работоспособности системы необходимо смоделировать архитектуру и передачу пакетов системы. Для этого необходимо выбрать средства моделирования архитектуры системы, разработки программного обеспечения и анализа работоспособности сети.

1.3.1 Обзор средств моделирования архитектуры

Основными параметрами для моделирования архитектуры системы является поддержка библиотек микроконтроллеров ATmega328 и ENC28J60. Рассмотрим несколько программ для автоматизации проектирования электроники.

1.3.1.1 Proteus

Система автоматизированного проектирования, позволяющая моделировать архитектуру электронных устройств. Программа состоит из нескольких модулей: ISIS и ARES. ISIS предназначен для редактирования электронных схем и

имитации их работы, а ARES предназначен для редактирования печатных плат. Кроме этого данное программное обеспечение оснащено встроенным редактором библиотек и автоматической системой размещения элементов на плате. Поддерживается возможность создания трехмерных моделей печатных плат [10]. Общий вид главного окна программы приведен на рисунке 3.

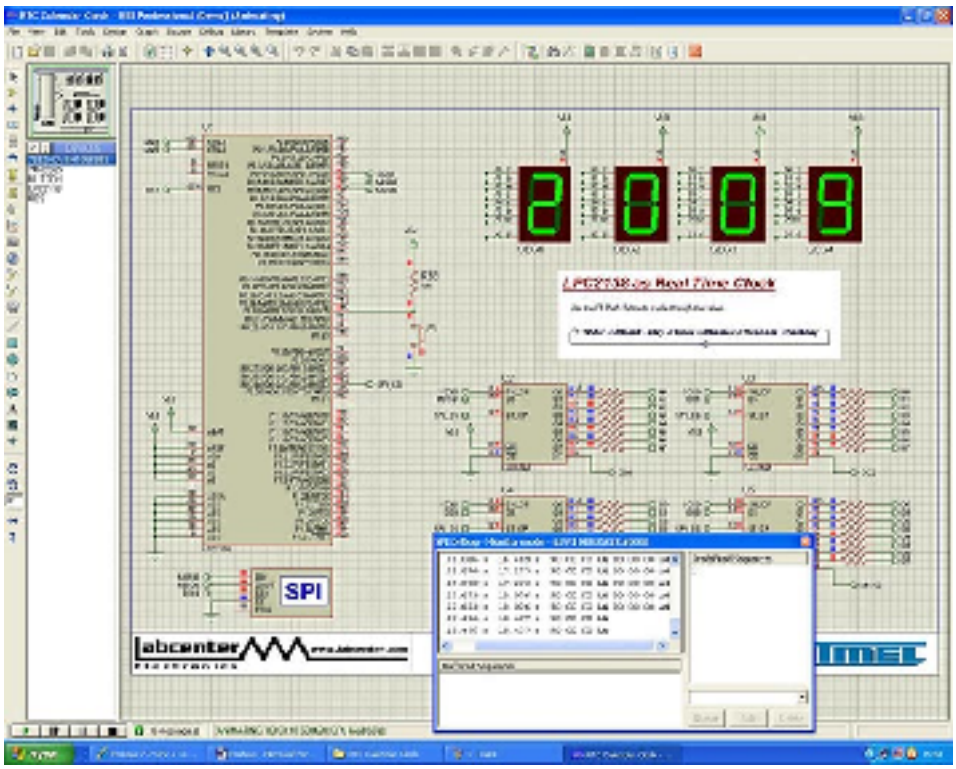


Рисунок 3 – Среда моделирования Proteus

Имеются инструменты, позволяющие подключать виртуальные устройства к USB и COM портам компьютера. В последующем, при подключении к этим портам внешних устройств, виртуальная схема работает с ними.

Proteus поддерживает несколько компиляторов: CodeVisionAVR, ICC, WinAVR. Для разработки доступны такие языки программирования как Assembler и C++. Используется в операционных системах Windows 2000/XP/Vista/7 и выше.

1.3.1.2 NI Multisim

Программный пакет для моделирования электронных схем и разводки печатных плат. Имеет очень простой интерфейс и мощные средства для анализа. Встроены виртуальные измерительные приборы с библиотеками реальных устройств. Взаимодействует со средой разработки систем измерения LabVIEW, что помогает сопоставлять теоретические данные с реальными во время создания печатных плат [11]. Общий вид главного окна программы приведен на рисунке 4.

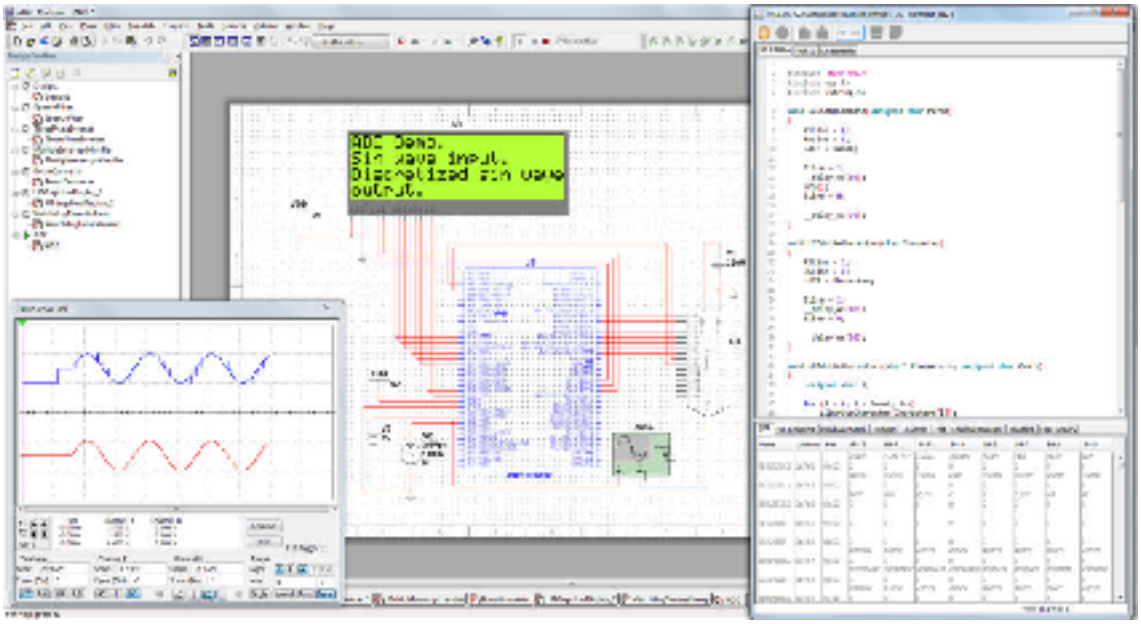


Рисунок 4 – Среда моделирования Multisim

Multisim поддерживает несколько языков программирования, например, Assembler и C++. Может использоваться в операционных системах 32 и 64 разрядов Windows XP/Vista/7 и выше.

1.3.1.3 LabVIEW

Графическая среда разработки для моделирования виртуальных приборов, которые состоят из лицевой панели, предназначенной для интерфейса с пользователем, и блок диаграммы, описывающей логику работы виртуального прибора. Для поддержки различных устройств в ПО LabVIEW включены дополнительные библиотеки для цифровой обработки сигналов, реализации баз данных, управления роботами, подключения внешнего оборудования по различным интерфейсам и протоколам [12]. Общий вид главного окна программы приведен на рисунке 5.

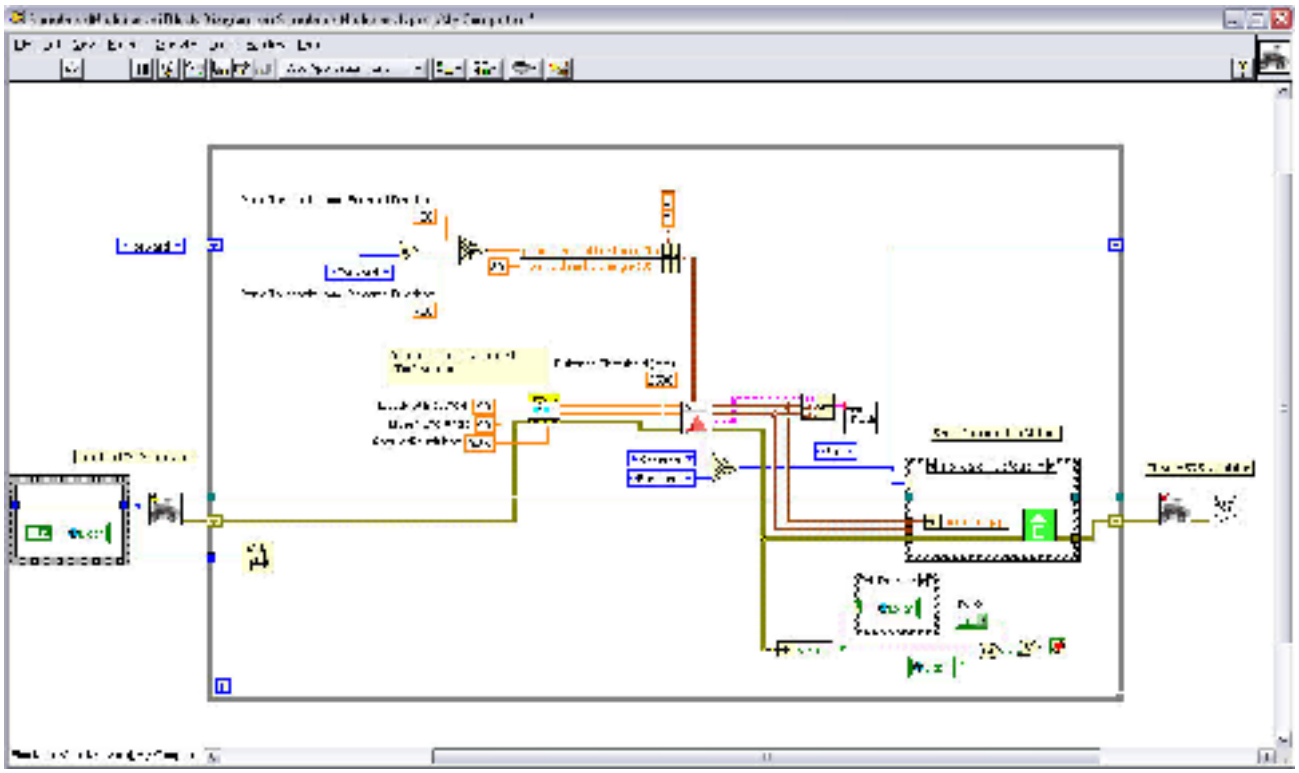


Рисунок 5 – Среда моделирования LabVIEW

LabVIEW поддерживает несколько языков программирования, например: C++, Pascal, Delphi. Используется в операционных системах Windows, MacOS, Linux.

1.3.2 Обзор средств разработки программного обеспечения

После моделирования архитектуры и сборки лабораторного макета необходимо написать программу для прошивки в микроконтроллер. Для этого проведен обзор сред разработки программного обеспечения с поддержкой языков программирования C/C++ и Assembler, а также поддержкой библиотек Microchip.

1.3.2.1 Atmel Studio

Среда разработки (IDE) для разработки приложений для 8- и 32-битных микроконтроллеров семейства AVR и 32-битных микроконтроллеров семейства ARM от компании Atmel. Для каждого микроконтроллера используется определенная разновидность языка Assembler, но их мнемоника достаточно схожа. Для работы с микроконтроллером (МК) необходимо изначально его проинициализировать, а затем настроить порты ввода и вывода, а также входы периферийных устройств [13]. Общий вид главного окна программы приведен на рисунке 6.

Atmel Studio содержит компилятор GNU C/C++ и эмулятор, позволяющий отладить выполнение программы без загрузки в микроконтроллер. Используется в операционных системах Windows NT/2000/XP/Vista/7/8/10.

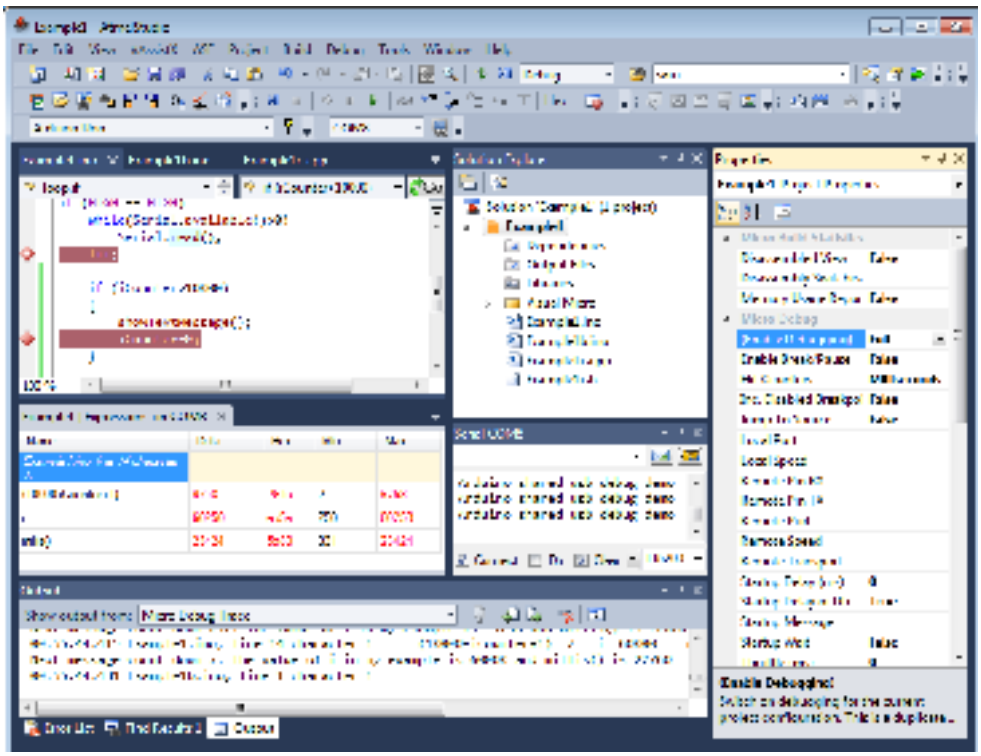


Рисунок 6 – Среда разработки Atmel Studio

1.3.2.2 Arduino IDE

Программная оболочка с текстовым редактором для программирования микроконтроллеров (например, ATmega328). Встроен компилятор AVR-GCC для программирования на языке C++ и библиотечные функции для упрощения программирования основных повторяющихся блоков [14]. Общий вид главного окна программы приведен на рисунке 7.

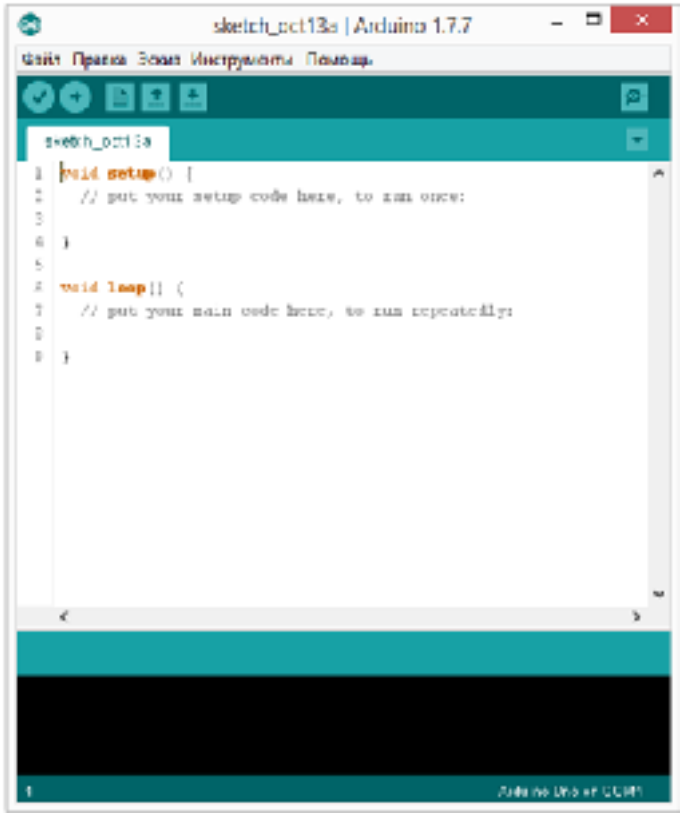


Рисунок 7 – Среда разработки Arduino IDE

1.3.2.3 CodeVisionAVR

Среда разработки для написания и отладки прикладных программ для микропроцессоров AVR [15]. Общий вид главного окна программы приведен на рисунке 8.

Работает в операционных системах семейства Windows. Включает в себя: компиляторы C и Assembler, генератор и редактор кода, а также модули взаимодействия с программатором Arduino Nano и отладочной платой STK-500.

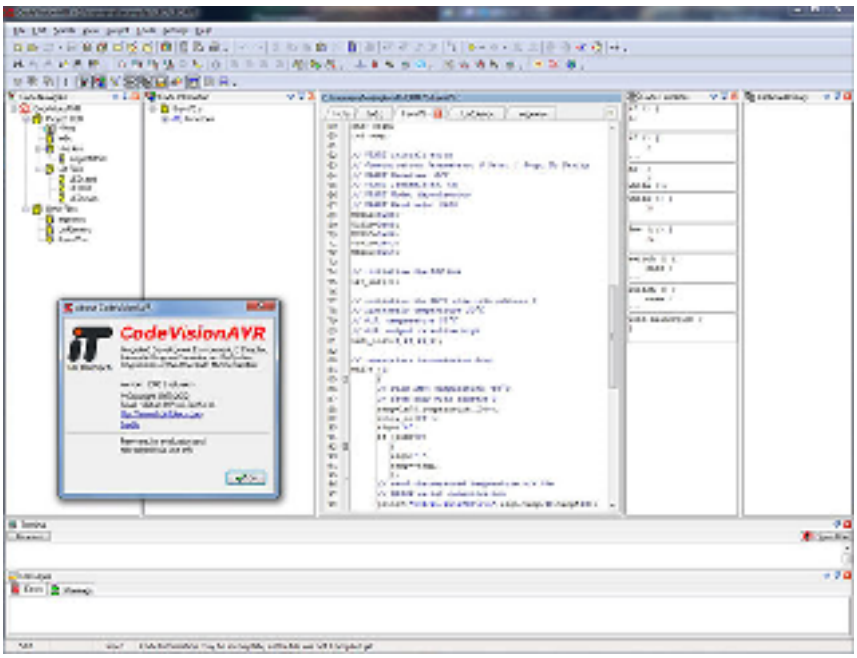


Рисунок 8 – Среда разработки CodeVisionAVR

1.3.3 Обзор средств анализа сети

Для проведения работоспособности системы необходимо выбрать ПО для моделирования и анализа трафика в сети.

1.3.3.1 Wireshark

Анализатор трафика сети. Содержит графический интерфейс для пользователя с возможностями сортировки и фильтрации информации. Позволяет просматривать поступающий трафик в режиме реального времени. Встроена поддержка библиотек для различных сетевых протоколов, что и позволяет расшифровывать пакет данных. Используется в Windows, Linux, Mac OS X, Solaris, FreeBSD, NetBSD [16]. Общий вид главного окна программы приведен на рисунке 9.

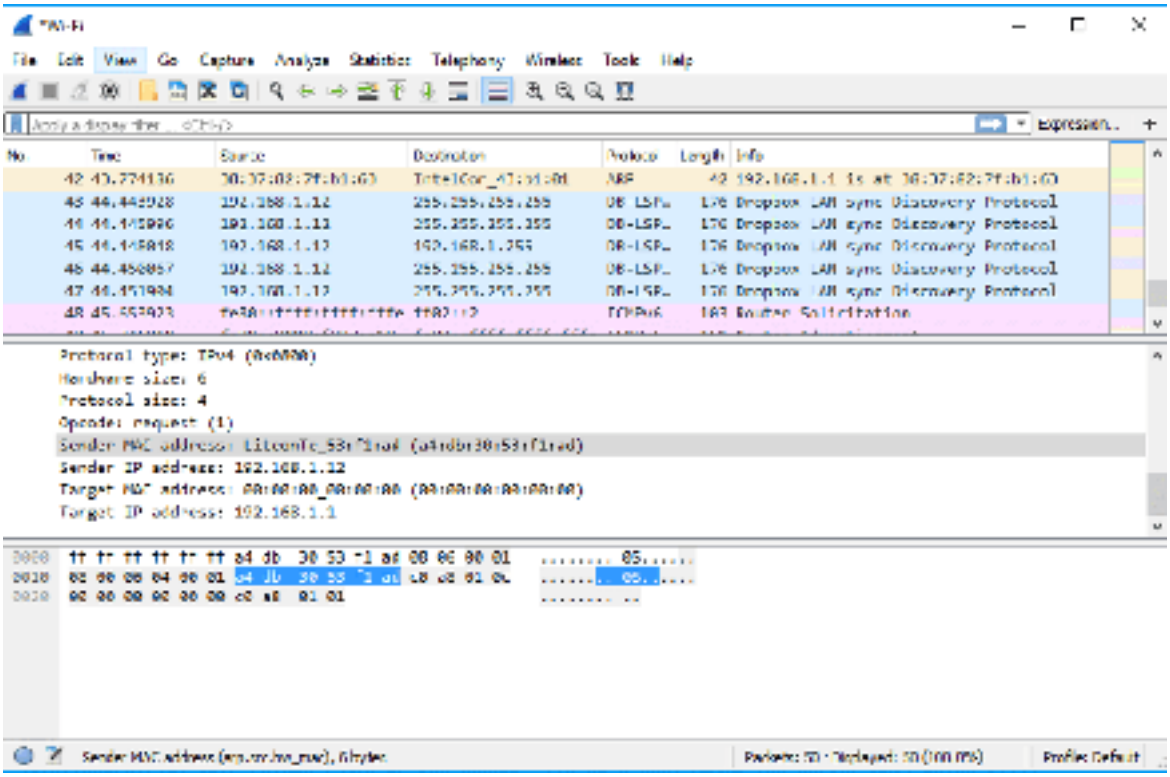


Рисунок 9 – Анализатор трафика Wireshark

1.3.3.2 NET-Simulator

Программное средство компании DRPA для моделирования поведения сети. Позволяет изменять топологию сети, пропускные способности каналов, нагрузку на сети, длины пакетов и распределение числа пакетов в одном сообщении. Есть возможность смоделировать маршрутизацию пакетов, различных методов и протоколов (RIP, OSPF, BGP и т.д.). Работает в операционных системах семейства Windows [17]. Общий вид главного окна программы приведен на рисунке 10.

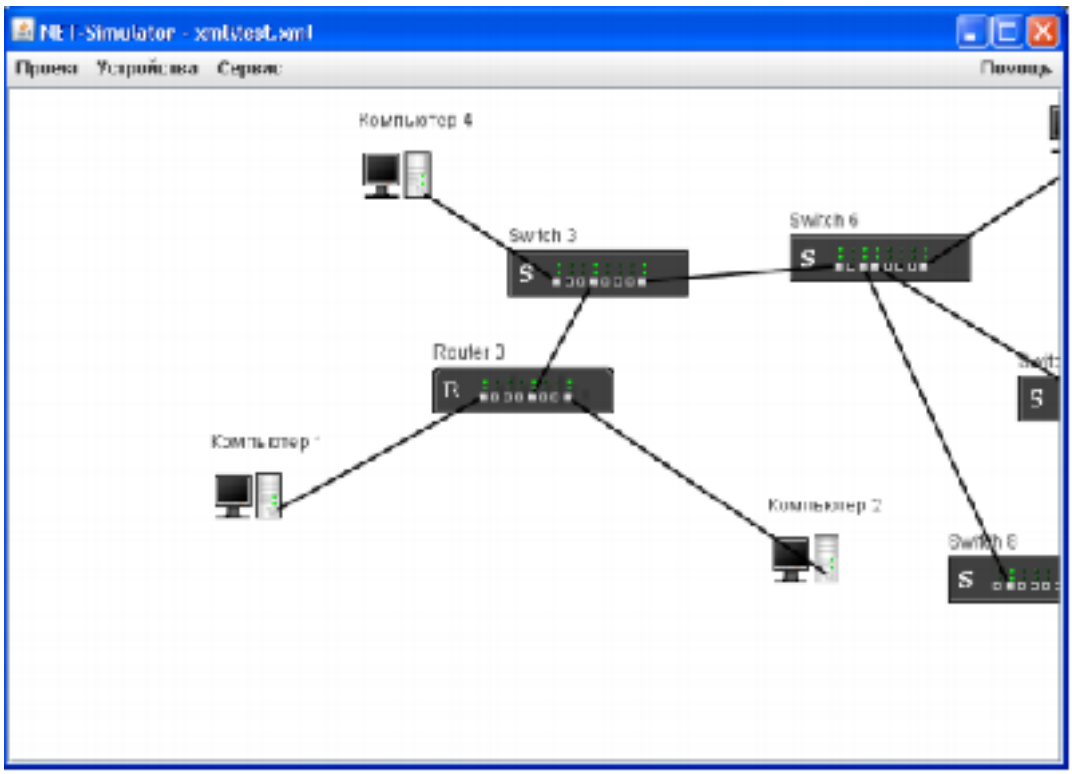


Рисунок 10 – Среда моделирования NET-Simulator

1.3.3.3 Pcap

Библиотека, предназначенная для анализа сети. Встраивается в различные симуляторы и анализаторы трафика, например Wireshark, tcpdump, Proteus. Используется совместно с языками C/C++, Java, .NET. Захватывает пакеты данных в сети с последующим сохранением, обработкой, чтением. Формат файла результата выполнения является .pcap. Используется в операционных системах семейства Windows как WinPcap, а в Linux как libpcap [18]. Общий вид главного окна программы приведен на рисунке 11.

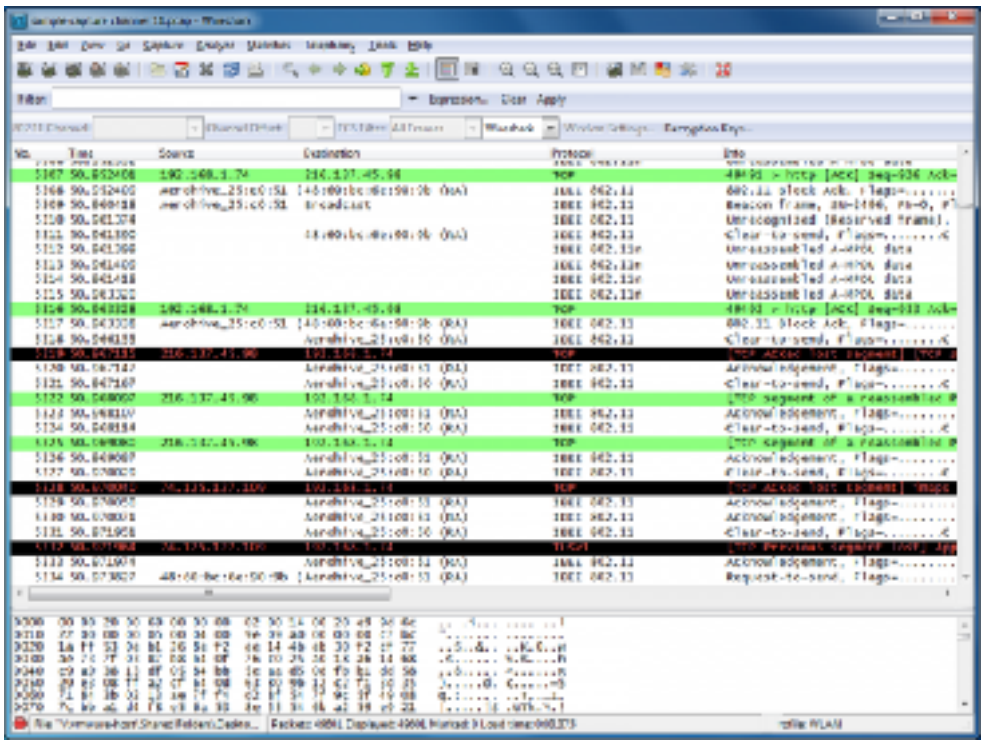


Рисунок 11 – Анализатор трафика Pcap

1.4 Обзор методов шифрования

Для реализации шифрованного канала связи, необходимо выбрать метод и вид шифрования, который будет реализован в ходе написания программного обеспечения.

Шифры – это методы и способы преобразования исходной информации с целью ее защиты от несанкционированного взлома и использования [19].

Шифрование подразделяют на несколько видов: симметричное и ассиметричное. Главным отличием симметричного кодирования является то, что при кодировании информации используется один и тот же ключ и для шифрования и для дешифрования. Во время ассиметричного шифрования может использоваться несколько ключей, включая закрытый ключ, которые располагаются у разных пользователей и используются однократно: для

шифрования, дешифрования. Таким образом, при отсутствии одного из ключей, расшифровать информацию невозможно.

Кроме того, симметричное шифрование чаще используется в системах с буферным хранилищем из-за того, что скорость криптографических операций намного выше, чем в ассиметричном шифровании. Следовательно, при реализации шифрованного канала связи следует использовать один из методов симметричного шифрования, для ускорения работоспособности системы.

В поставленных задачах на выполнение ВКР выделено 3 метода шифрования, из которых будет выбран способ кодирования информации из пакета. Это шифр Цезаря, таблица Виженера и квадрат Полибия. Все эти методы относятся к шифрованию с помощью замены.

1.4.1 Шифр Цезаря

Один из самых простых и широко известных методов. Шифр назван в честь римского императора Гая Юлия Цезаря, который использовал данный метод для шифрованной переписки со своими генералами [19]. Особенностью данного метода является шифрование алфавита с помощью подстановок букв из сдвинутого алфавита. Сдвиг осуществляется вправо на определенное количество символов, например на 3 символа. В результате формируется новый шифрованный алфавит. Например, как показано на рисунке 12, символ «B» заменяется на «E».

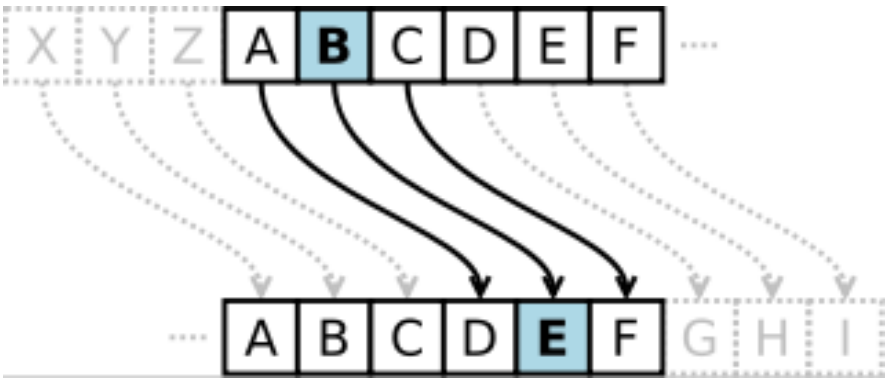


Рисунок 12 – Шифр Цезаря со сдвигом на 3

Для примера зашифруем фразу «methods» со сдвигом на 3 символа (таблица 3).

Таблица 3 – Пример шифрования шифром Цезаря

Исходные данные	m	E	t	h	o	d	s
Результат шифрования	p	H	w	k	r	g	v

Из-за того, что данный шифр очень прост в реализации, то его очень легко взломать. Зная зашифрованный текст и то, что использован метод шифрования Цезаря, злоумышленник, с помощью подбора числа сдвига, может расшифровать информацию.

1.4.2 Таблица Виженера

Считается усложненным шифром Цезаря, за счет того, что состоит из нескольких шифрованных алфавитов, с разными значениями сдвига, располагающихся в центральной части таблицы. Применяется в основном с таблицами латинского алфавита. Тогда таблица Виженера составляется из строк по 26 символов, и каждая следующая строка сдвигается на один символ (рисунок 13). По краям таблицы располагаются алфавиты, предназначенные для ключевого слова (вертикальный) и данных (горизонтальный) [16].

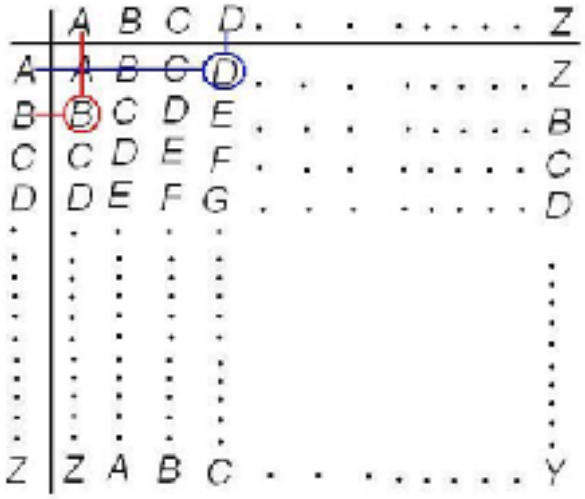


Рисунок 13 – Таблица Виженера, пример шифрования

Для примера зашифруем фразу «methods» с помощью таблицы Виженера с ключом «car» (таблица 4).

Таблица 4 – Пример шифрования таблицей Виженера

Исходные данные	m	e	t	h	o	d	s
Кодовое слово	c	a	r	c	a	r	c
Результат шифрования	o	e	k	j	o	u	u

Реализовать данный метод можно при обработке массива. В таком случае кодовое слово генерируется и хранится в прошивке.

1.4.3 Квадрат Полибия

Данный метод представлен обычной таблицей, которая формируется с нумерованием строк. Вовнутрь таблицы заносятся все символы, которые шифруются (рисунок 14). Существует несколько видов шифрования данным методом: замена биграммами и замена координатами [19].

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I	J
3	K	L	M	N	O
4	P	Q	R	S	T
5	U	V	W	X	Y
6	Z	1	2	3	4
7	5	6	7	8	9
8	0	.	,	?	!

Рисунок 14 – Квадрат Полибия

При кодировании с помощью биграмм, символ исходного текста меняется на нижестоящий символ в таблице. Пример рассмотрен в таблице 5.

Таблица 5 – Пример шифрования квадратом Полибия

Исходные данные	m	E	t	h	o	d	s
Результат шифрования	r	J	y	m	t	i	x

При шифровании с помощью замены координат, символ исходного текста меняется на координаты его расположения в таблице. Пример рассмотрен в таблице 6.

Таблица 6 – Пример шифрования квадратом Полибия

Исходные данные	m	E	t	h	o	d	s
Координата горизонтальная	3	1	4	2	3	1	4
Координата вертикальная	3	5	5	3	5	4	4
Результат шифрования	33	15	45	23	35	14	44

Выводы к Главе 1

1. Произведен анализ существующих решений сетевых контроллеров, в результате которого для реализации системы выбран Ethernet Shield на базе микроконтроллера ENC28J60.

2. Для разработки программы выбрана среда разработки с возможностью подключения программатора Arduino Nano и поддержкой языков программирования Assembler, C/C++.

3. Выбраны средства моделирования работоспособности системы. Для программно-аппаратной реализации архитектуры – Proteus, а для проверки правильности работы сети и анализа трафика – Wireshark.

4. В результате анализа методов шифрования, выбрана таблица Виженера как наиболее устойчивый шифр от несанкционированного доступа злоумышленников. Кроме того, данный способ достаточно легко реализовать при разработке программного обеспечения.

2 Архитектура и программная модель контроллера с защищенным каналом Ethernet

Для разработки лабораторного стенда и программного обеспечения необходимо определить структурный и функциональный состав системы.

2.1 Описание структурной схемы

Разработанная структурная схема лабораторного стенда представлена на рисунке 15. Для реализации стенда используются выбранные ранее микроконтроллер ATmega328 и Ethernet Shield на базе микросхемы ENC28J60.

Подключение микроконтроллера к Ethernet модулю происходит через контактный разъем типа PHDL, а именно SPI интерфейс (таблица 7). Загруженная через него прошивка необходима для программного тестирования работоспособности сети точка-точка. Сетевой модуль подключается к каналу Ethernet, через который реализуется получение трафика.

Таблица 7 – Назначение контактов Ethernet модуля

Контакт	Обозначение	Назначение	Направление сигнала
1	CLK	Выход тактового сигнала	Выход
2	INT	Сигнал прерывания	Выход
3	WOL	Зарезервированный контакт	-
4	SO	Сигнал SO интерфейса SPI	Выход
5	SI	Сигнал SI интерфейса SPI	Вход
6	SCK	Сигнал SCK интерфейса SPI	Вход
7	CS	Сигнал CS интерфейса SPI	Вход
8	RST	Сигнал сброса	Вход
9	VCC	Питания модуля 3.3 В, 180 мА	-
10	GND	Общий вывод, земля	-

Кроме того, микроконтроллер подключен к компьютеру, через который реализуется его программирование. Питание на микроконтроллер подается 3.3 В с

выходным током 180 мА, для того, чтобы запитать все драйвера передатчика и микросхему [20].

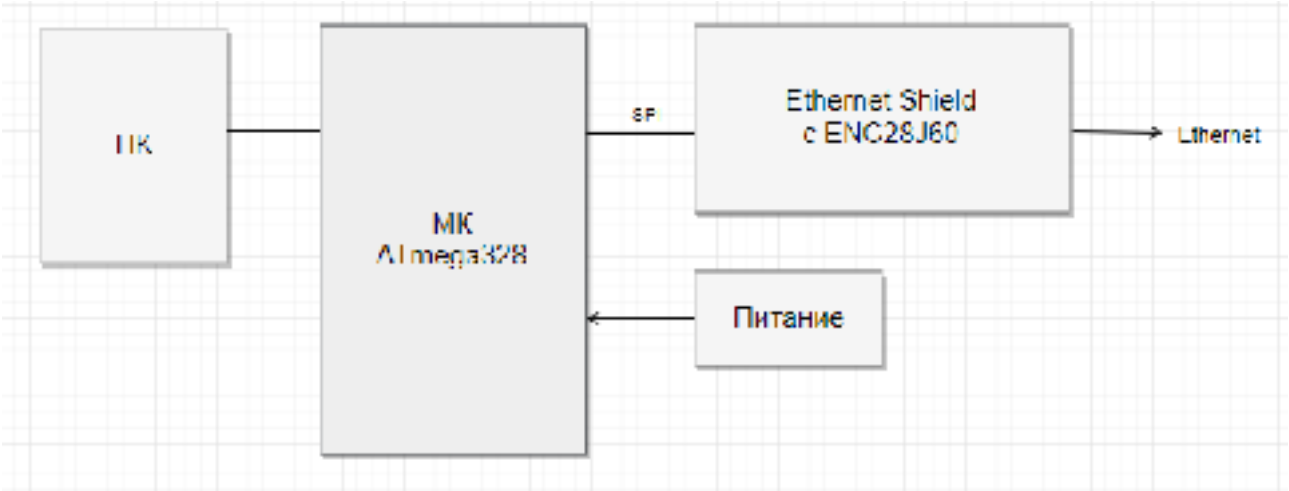


Рисунок 15 – Структурная схема архитектуры лабораторного стенда

2.2 Разработка функциональной схемы

На рисунке 16 изображены функциональные блоки разработанного лабораторного стенда: ПК, питание микроконтроллера, микроконтроллер, Ethernet модуль, канал связи стандарта Ethernet. Рассмотрим отдельно эти составляющие.

Для работы микроконтроллера используется питание, подключенное к положительному входу напряжения Vcc. Как уже упоминалось ранее, используется напряжение в 3.3 В при частоте 6 МГц.

Компьютер подключен через переходник USB – USB-Mini к программатору Arduino Nano, на котором располагается микроконтроллер ATmega328.

Ethernet модуль подключен к программатору через интерфейс SPI. Это необходимо для того, чтобы запрограммировать сетевой модуль на прием и передачу кадров. При подключении SPI используется пять контактов на микроконтроллере (I/O, SDO, SDI, SCK, INTx) и на сетевом модуле

(инвертированный вход CS, SI, SO, SCK, инвертированный выход INT), которые описаны ранее в таблице 3.

Ethernet модуль состоит из 3 основных программных блоков:

Buffer – оперативная память с двумя портами, позволяющая одновременно выполнять несколько операций чтения или записи. Реализует хранения прошивки и встроенных логических операций. Объем памяти составляет 8 Кбайт.

MAC – интегрированная функциональная возможность модуля. Определяет MAC-уровень для работы с сетью, что уменьшает нагрузку на управляющий контроллер. Соединен с памятью и PHY.

PHY – физический интерфейс, подключенный к MAG-Jack, который состоит из двух трансформаторов (Ethernet transformer), разъема RJ-45 и двух светодиодов (LEDA, LEDB). Фильтры необходимы для развязки и защиты от статического потенциала с Ethernet кабеля [21]. Интерфейс RJ-45 используется для подключения к каналу связи Ethernet, а светодиоды используются как индикаторы, отвечающие за оповещение о работе сети.

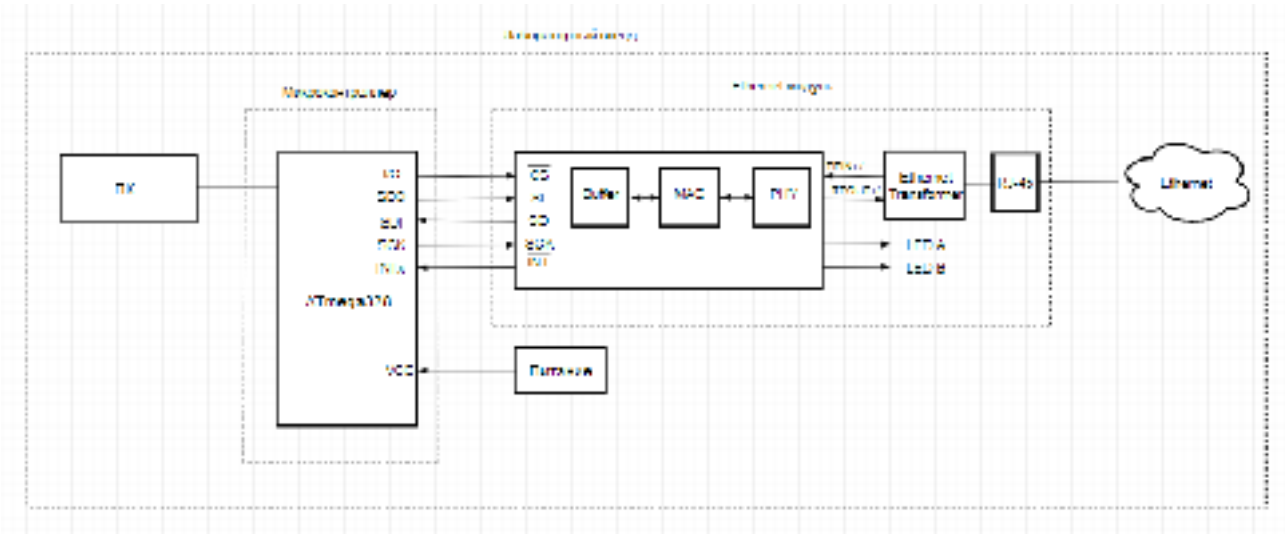


Рисунок 16 – Функциональная схема лабораторного стенда

2.3 Выбор состава оборудования

Определен состав оборудования для лабораторного стенда:

1. Персональный компьютер.
2. Программатор Arduino Nano на базе микроконтроллера ATmega328.
3. Ethernet Shield на базе микросхемы ENC28J60.
4. Интерфейсный кабель-преобразователь USB – USB-Mini.
5. Однопиновые соединительные кабели типа мама-мама – 8 штук.
6. Патч-корд – 2 штуки (витая пара, прямой кабель).
7. Программное обеспечение Arduino IDE.

На основе выбранного оборудования собран лабораторный стенд (рисунок 17), в котором используется полудуплексный канал приема/передачи пакетов.

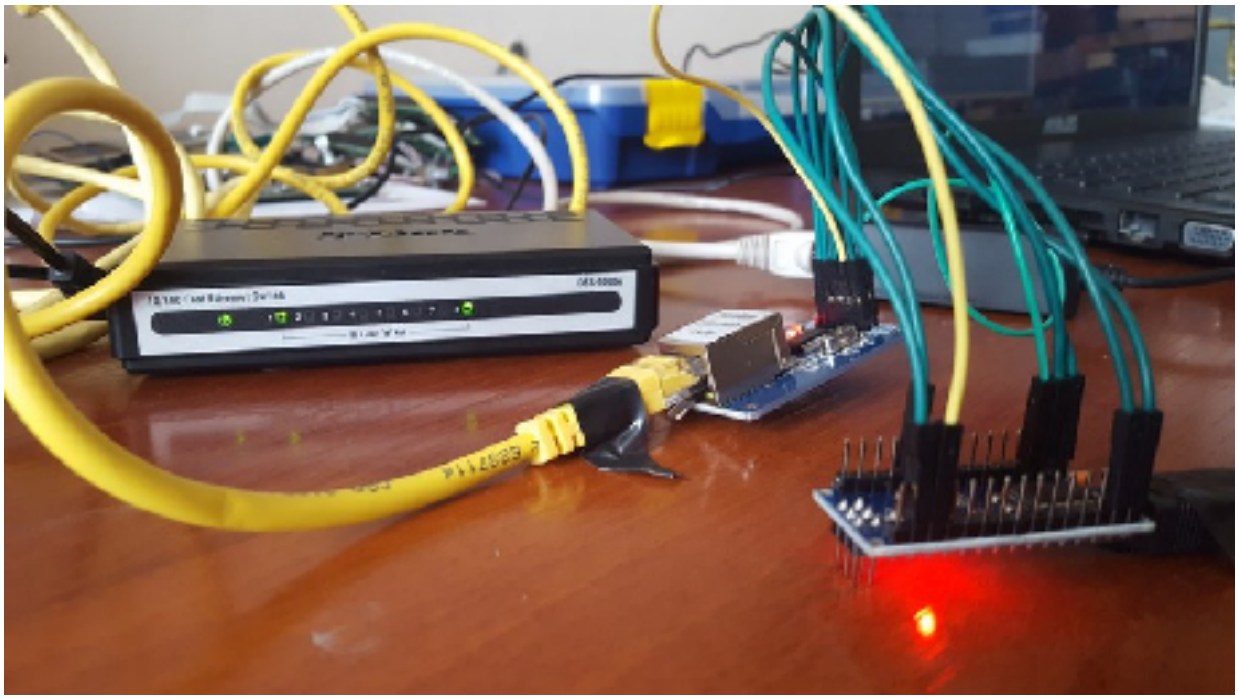


Рисунок 17 – Лабораторный стенд

Выводы к Главе 2

Разработаны структурная (рисунок 15) и функциональная (рисунок 16) схемы лабораторного стенда. Выполнены анализ, поиск и выбор элементов для разработки лабораторного стенда. Все используемые элементы, предварительно протестированы. На основе работоспособных модулей собран лабораторный стенд.

Проведено предварительное тестирование лабораторного стенда на его работоспособность при включении питания, а также протестированы функции сетевого контроллера. Функционирование стенда происходит в штатном режиме, в том числе при программировании и начальном старте стенд выходит на рабочий режим, о чем сигнализируют светодиоды на разъеме MAG-Jack. В режиме связи с персональным компьютером стенд программируется корректно (рисунок 21).

3 Программное и аппаратное обеспечение лабораторного прототипа

Для проверки работоспособности лабораторного стенда необходимо разработать программное обеспечение. Для отработки технических решений, согласно заданию на ВКР, необходимо реализовать обмен «точка-точка» и режимы приема и передачи пакетов. Именно поэтому необходимо определить алгоритмы приема и передачи пакетов и реализовать их программное решение.

3.1 Архитектура программного обеспечения

При разработке архитектуры программного обеспечения, применен принцип каркасной сборки, что позволило предоставить необходимый инструментарий в удобной форме. Таким образом, не требуется изменять все программное обеспечение, а только редактировать необходимые модули. Например, модуль шифратора (рисунок 18).

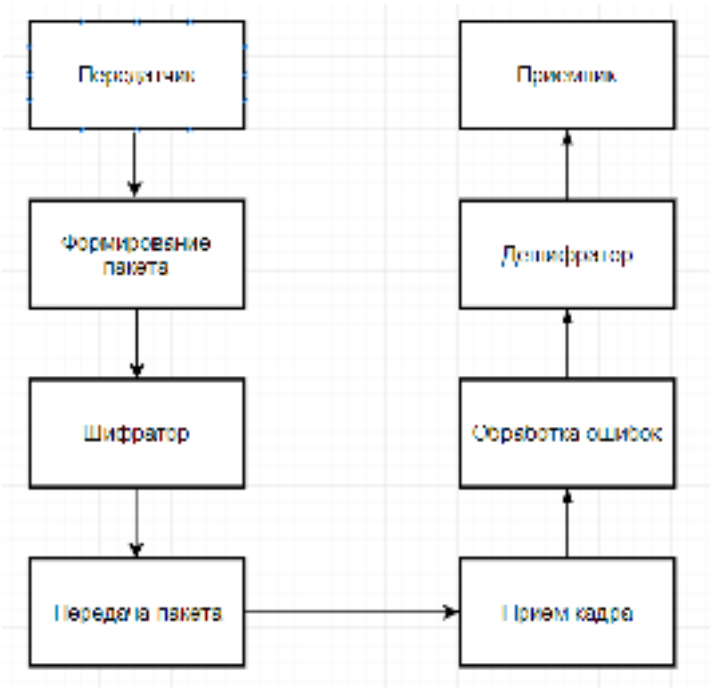


Рисунок 18 – Схема архитектуры программного обеспечения

3.2 Разработка алгоритмов приема и передачи пакетов

Для разработки алгоритма работы сети, необходимо определить возможные состояния при этом, а именно при приеме и передаче пакетов.

3.1.1 Передача пакетов

На рисунке 19 изображена диаграмма состояний при передаче пакетов. В узлах находятся состояния системы, а линии связи обозначают процесс для достижения определенного состояния. Разберем диаграмму пошагово, начиная из состояния «Начало отправки»:

1. Происходит передача преамбулы, которая обеспечивает синхронизацию приемника и разграничивает кадры между собой. Занимает 8 байт.

2. Передается октет SFD, который отделяет полезные данные от остального пакета. Занимает 6 байт.

3. Передается сформированный заголовок с MAC приемника и передатчика. Занимает 2 байта.

4. Происходит передача данных из памяти, организованной по принципу First In First Out (FIFO). Осуществляется до тех пор, пока данные не закончатся в FIFO, либо до максимального размера кадра (от 46 до 1500 байт).

5. Производится подсчет контрольной суммы пакета и передача ее по каналу.

6. Весь пакет передан. Осуществляется завершение отправки пакета.

7. Ожидание межкадрового интервала (IPG). Составляет 96 битовых последовательностей, что занимает около 9,6 микросекунды в сети Ethernet.

8. Ожидается поступление данных на передачу.

9. Появились данные для передачи. Происходит запись в память FIFO.

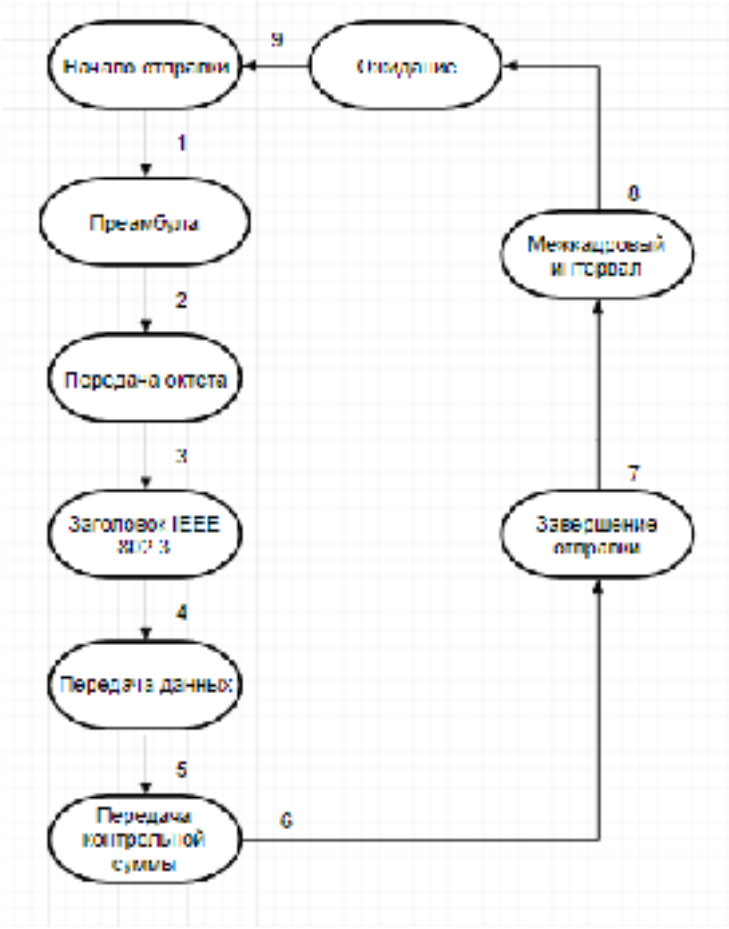


Рисунок 19 – Диаграмма состояний передачи пакета

3.1.2 Прием пакетов

На рисунке 20 изображена диаграмма состояний при приеме пакетов. В узлах находятся состояния системы, а линии связи обозначают процесс для достижения определенного состояния. Разберем диаграмму пошагово, начиная из состояния «Начало приема»:

1. Осуществляется прием октетов 0x55 до 0xd5. Это и есть преамбула, которую отправляли в передатчике [1].

2. Принимается октет 0xd5, который отделяет полезные данные от остального пакета.

3. При ошибке приема октетов преамбулы вызывается ошибка.

4. Осуществляется прием данных до тех пор, пока есть что принимать, либо пока не закончится максимально возможный размер кадра (от 46 до 1500 байт).

5. При ошибке приема октета 0xd5 вызывается ошибка.

6. Подсчитывается и проверяется контрольная сумма пакета. Она должна сойтись с той, которая находится в пакете.

7. Если превышен размер принимаемого пакета, прекращается анализ кадра.

8. Если возникли проблемы с приемом данных, вызывается ошибка.

9. Пакет успешно получен. Завершается прием пакета.

10. Вычисленная контрольная сумма не соответствует той, которая получена из пакета. Вызывается ошибку.

11. Ранее (шаг 7) определено, что пакет превысил максимальный размер. Вызывается ошибка.

12. Вызвана ошибка. Завершается прием пакета.

13. Ожидание межкадрового интервала (IPG). Составляет 96 битовых последовательностей, что занимает около 9,6 микросекунды в сети Ethernet.

14. Ожидается поступления данных на прием.

15. Поступили данные на прием. Начинаем принимать пакет.

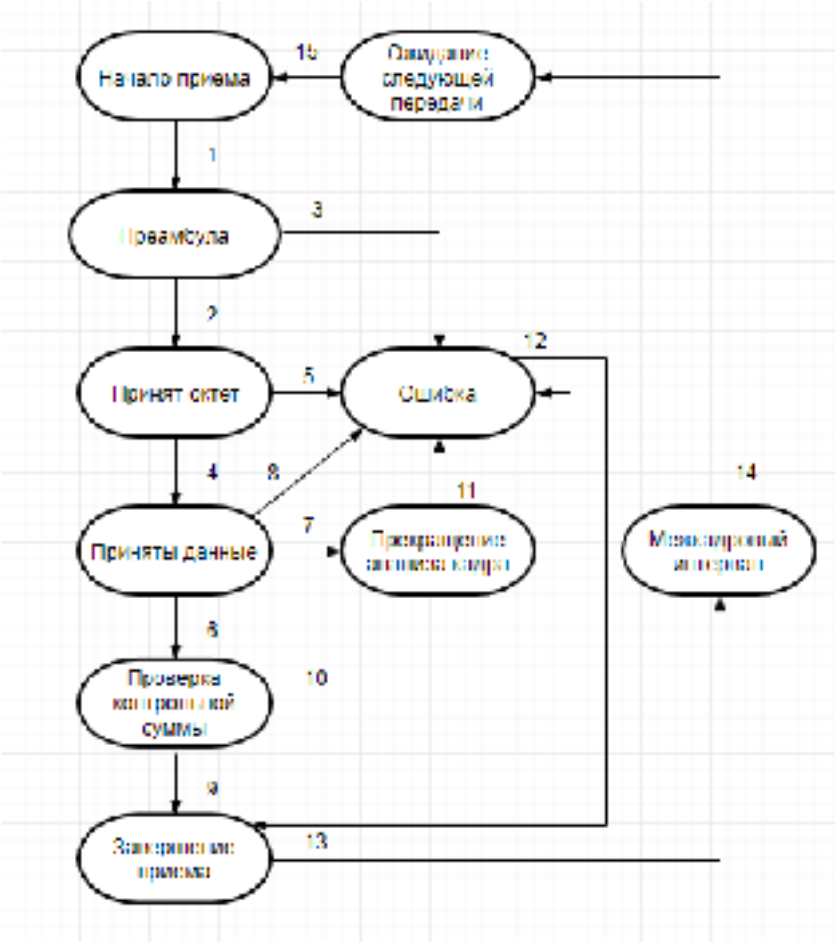


Рисунок 20 – Диаграмма состояний приема пакета

3.2 Разработка программного обеспечения

По представленному в пункте 3.1 алгоритму разработано программное обеспечение, которое осуществляет прием, передачу пакетов в сети точка-точка и их шифрование с помощью алгоритма таблицы Виженера.

На рисунке 21 показано количество использованной памяти при загрузке разработанного программного обеспечения через Arduino Nano в Ethernet Shield. Во время разработки и доработки ПО использовалось не более 15% от общего объема памяти устройства.

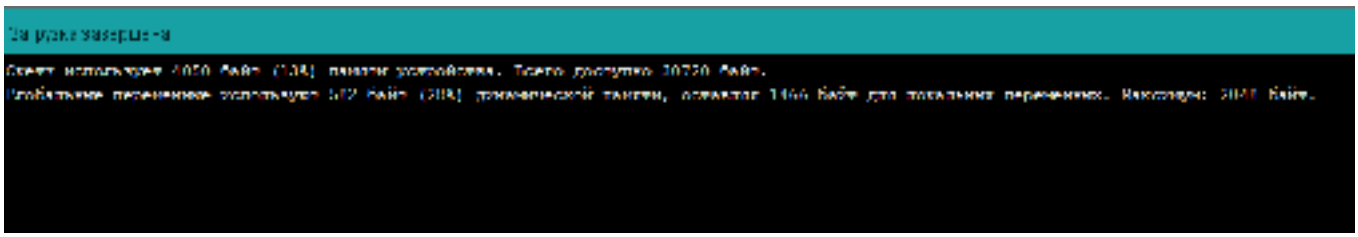


Рисунок 21 – Уровень загрузки памяти в Arduino Nano

Выводы к Главе 3

Разработаны и описаны диаграммы состояний для приема (рисунок 19) и передачи (рисунок 20) пакетов. Разработано программное обеспечение, реализующее формирование, прием, передачу пакетов (листинг программы приведен в ПРИЛОЖЕНИИ Б). Реализовано шифрование данных с помощью алгоритма таблицы Виженера.

4 Результаты тестирования и сравнительного анализа полученных решений

4.1 Отработка модулей в средах моделирования

Для тестирования модулей сетевого контроллера проведено их моделирование в программном обеспечении Proteus. На рисунке 22 представлена схема во время моделирования. На ней располагаются: программатор (SIM1), микросхема с поддержкой стандарта IEEE 802.3 (ENC28J60), набор потенциометров, виртуальный терминал.

Для проверки работоспособности каждого модуля в отдельности, разработан тестовый код, реализующий работу сервера на сетевом контроллере. Программное обеспечение загружается в модуль программатора и через SPI интерфейс (SCK, MOSI, MISO, CS) передает необходимые настройки в микросхему ENC28J60. Входные данные подаются на аналоговый вход программатора и потенциометры.

В результате выполнения программы загорается зеленый светодиод (в структуре Link), который сигнализирует о том, что устройство работает корректно. А на виртуальном терминале отображается IP адрес разработанного сервера.

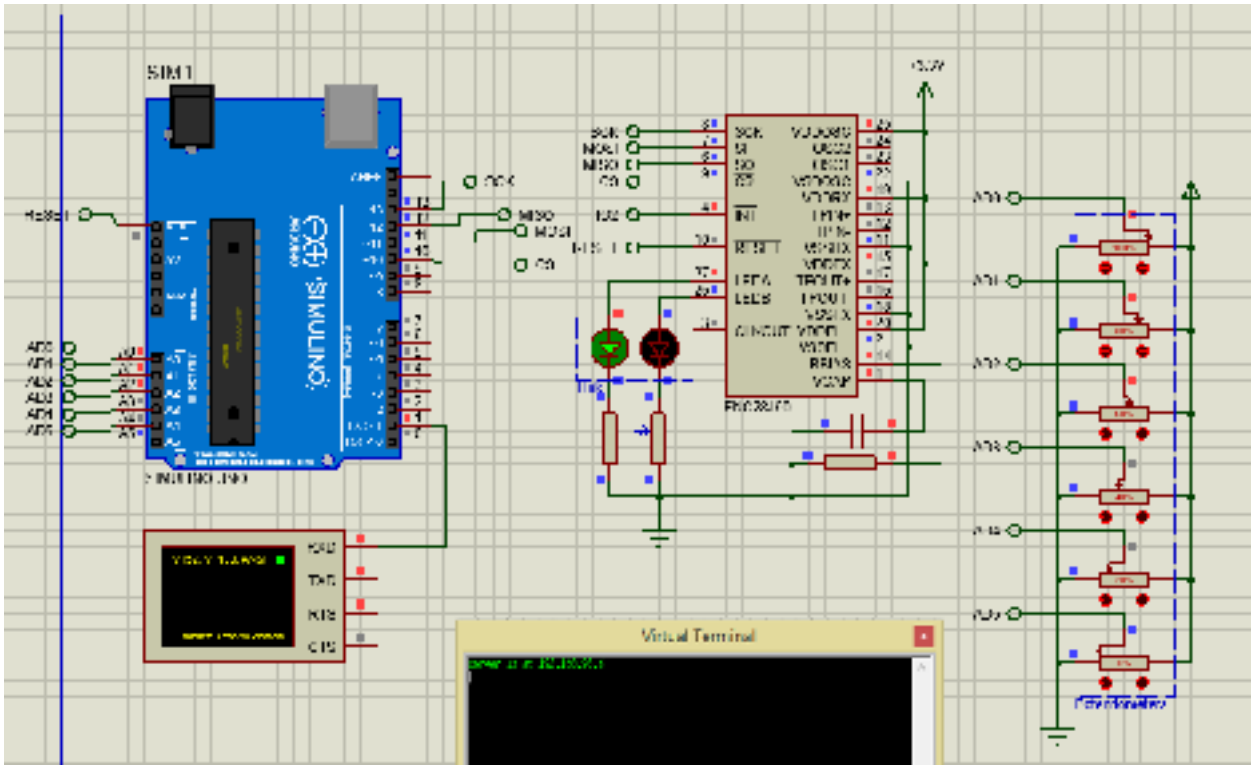


Рисунок 22 – Схема во время моделирования в программном обеспечении Proteus

4.2 Проверка работоспособности программного обеспечения

При тестировании разрабатываемого программного обеспечения выполнены следующие тестовые решения:

1. Проведена отладка канала связи при передаче пакетов тремя способами (рисунок 23).

2. Разработана тестовая веб-страница, которая формируется на сетевом контроллере и передает информацию в виде HTML страницы на пользовательский ПК (рисунок 27).

Разберем каждую проверку подробнее.

На рисунке 23 показаны способы тестовых передач пакетов:

1. Произведена стандартная отправка пакетов от передатчика к приемнику и обратно с использованием разработанного программного обеспечения (рисунок

25) и без (рисунок 24), для тестирования скорости передачи данных по каналу связи Ethernet. Проведена проверка программного обеспечения на корректность формирования пакета в сетевом контроллере, его протоколов приема и передачи (рисунок 26).

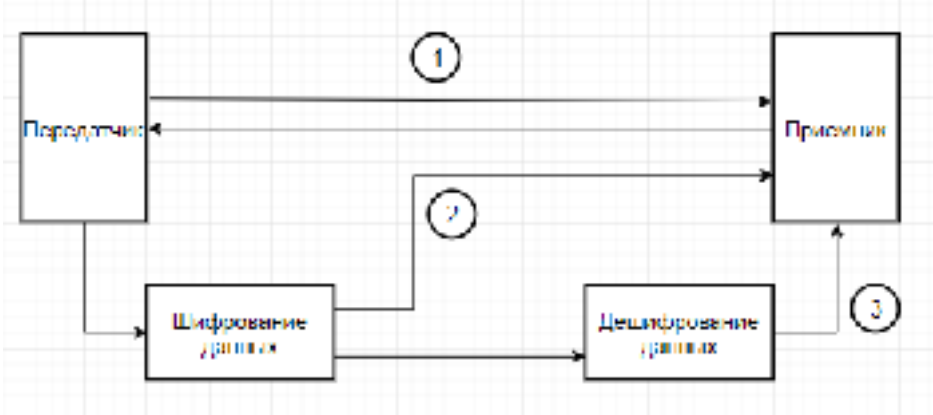


Рисунок 23 – Способы отладки канала связи

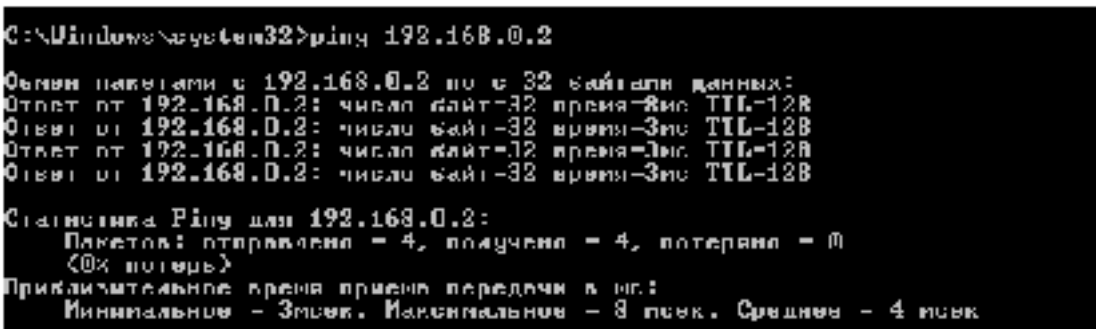


Рисунок 24 – Проверка работоспособности сети при передаче данных без использования разработанного программного обеспечения

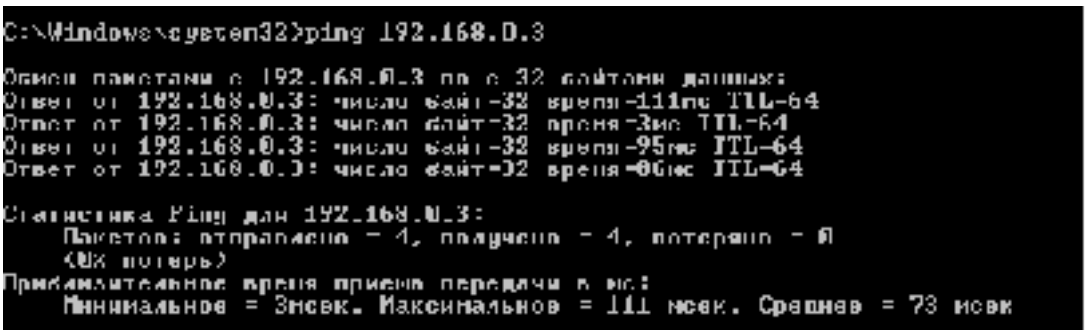


Рисунок 25 – Проверка работоспособности сети при передаче данных с использованием разработанного программного обеспечения

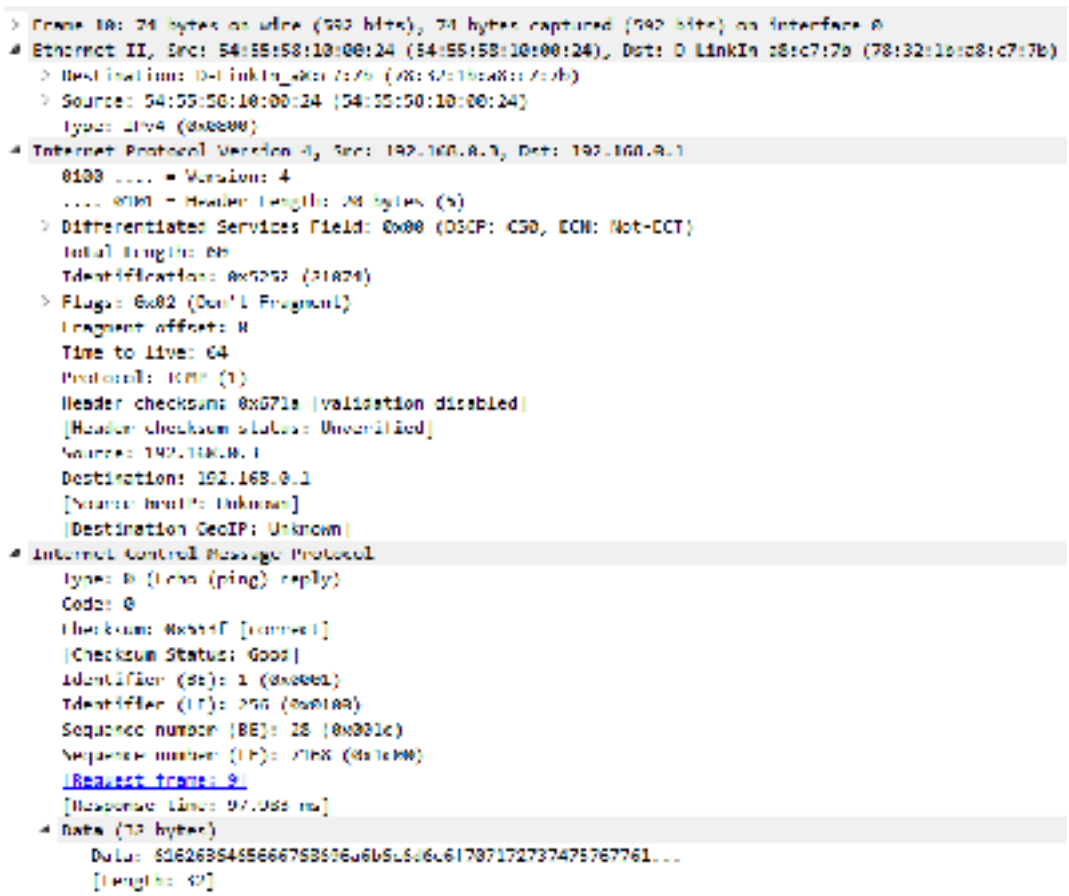


Рисунок 26 – Сформированный пакет

2. Отправка пакетов от передатчика к приемнику с включением в программное обеспечение шифрования данных. В этом случае возникает ошибка при приеме данных. Пакет доходит в корректном шифрованном виде, но приемник не может распознать его структуру из-за того, что не происходит дешифрация данных (рисунок 27).

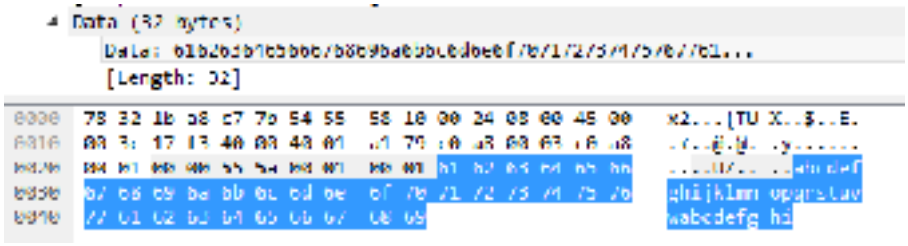


Рисунок 27 – Вид данных в пакете при передаче

3. Отправка пакетов от передатчика к приемнику через полноценный защищенный канал Ethernet с шифрованием и дешифрованием данных. В этом случае пакет успешно принимается из-за того, что проходит полный цикл с шифрацией и дешифрацией информации из пакета.

Следующим видом тестирования является разработка тестовой веб-страницы, которая формируется на сетевом контроллере и передает информацию с использованием протокола HTTP на пользовательский ПК. На рисунке 28а изображен изначальный вид тестовой страницы, а на рисунке 28б вид пакета, полученный при выводе этой страницы.

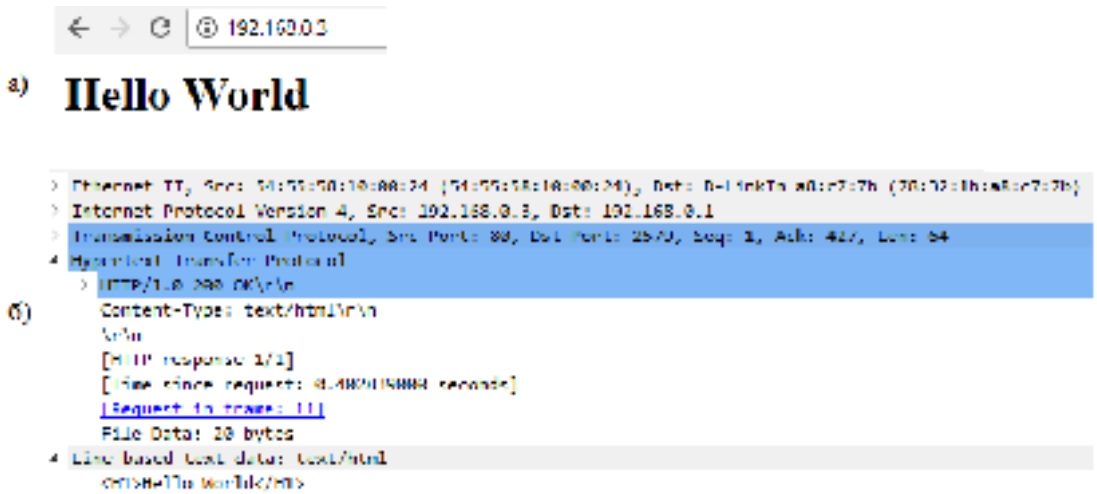


Рисунок 28 – а) вид тестовой страницы, б) вид пакета с информацией, которая выводится на страницу

Далее были добавлены кнопки управления, для изменения вида страницы и данных пакета (рисунок 29а). К примеру, при нажатии на кнопку «Open» в URL страницы появляется информация о перенаправлении на «/start» (рисунок 29б), а при нажатии на кнопку «LED IS ON» – «LED» (рисунок 29в).

Рисунок 29 – а) вид веб-страницы при загрузке, б) вид веб-страницы после нажатия на кнопку «Open», в) вид веб-страницы после нажатия на кнопку «LED IS ON»

Помимо изменений, которые происходят на веб-странице, в виде графического представления для пользователя, вносятся изменения в HTTP заголовке поля Request URL (рисунок 30). При загрузке стартовой страницы ничего не происходит (рисунок 30а). Если нажата одна из кнопок «Open» или

«LED IS ON», то при формировании пакета вносятся изменения в URL в виде «/start» (рисунок 30в) или «/LED» (рисунок 30б) соответственно.

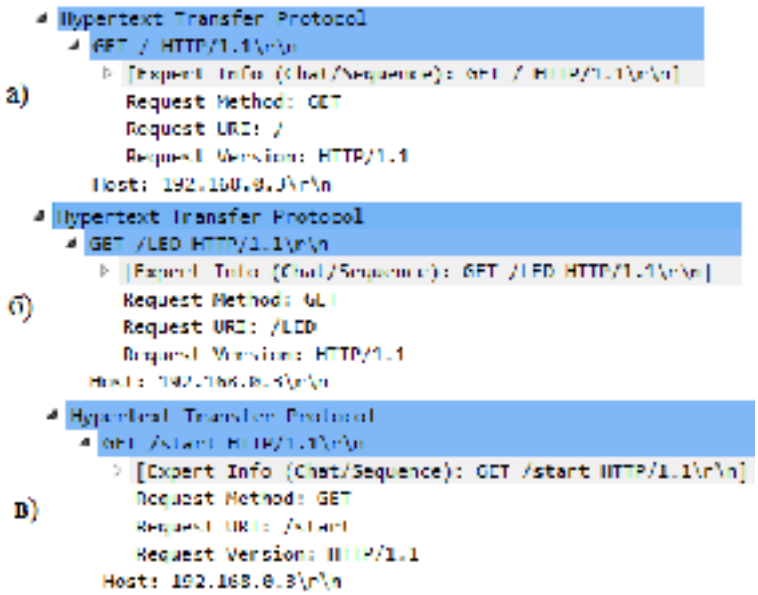


Рисунок 30 – а) вид пакета при загрузке веб-страницы, б) вид пакета после нажатия на кнопку «LED IS ON», в) вид пакета после нажатия на кнопку «Open»

4.3 Анализ трафика

После тестирования программного обеспечения проведен анализ входящего и исходящего трафика. При различных запросах и пересылаемых данных очередность переданных пакетов соответствует правилам передачи по транспортному уровню протокола TCP (рисунок 31).

1. Изначально происходит поиск адреса сервера (192.168.0.3) в сети.

2. Далее пользователь (192.168.0.1) отправляет запрос (SYN) серверу на установление соединения и ожидает ответа.

3. Сервер получает запрос от пользователя на соединение, отправляет ответный запрос на подтверждение (SYN, ACK) и переходит в состояние SYN-RECEIVED.

4. Пользователь получает запрос от сервера и подтверждает его флагом ACK.

5. Происходит обмен данными между сервером и пользователем при помощи GET запросов и протокола передачи гипертекста HTTP.

6. При появлении флага FIN сервер перестает передавать данные пользователю и завершает соединение.

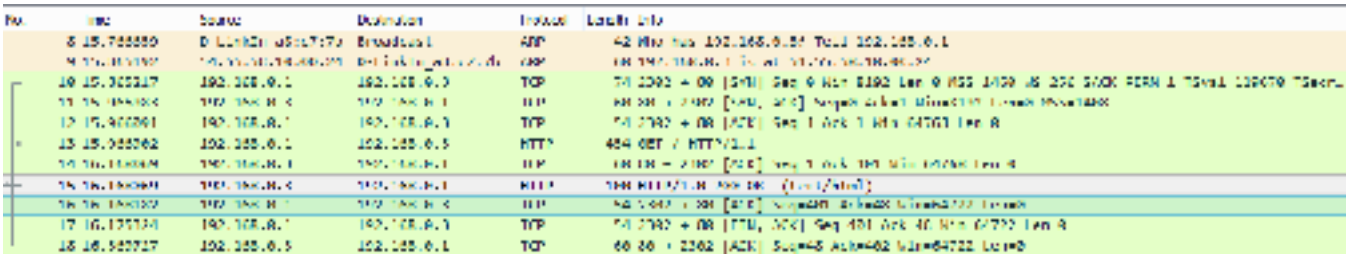


Рисунок 31 – Последовательность передаваемых пакетов при работе тестовой веб-страницы

Выводы к Главе 4

При тестировании лабораторного стенда выяснилось, что разработанное программное обеспечение работает корректно. Обеспечивает правильную передачу пакетов по транспортному уровню, формирует пакеты в соответствии со стандартом Ethernet и далее встраивает их в пакет передачи TCP.

Реализованное шифрование работает последовательно. Без дешифрования пакета его невозможно прочитать, либо он не опознается приемником.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы разработан реконфигурируемый сетевой контроллер с шифрованным каналом Ethernet. В пояснительной записке описаны программно-аппаратный инструментарий для разработки такой системы, а также ее архитектура и этапы разработки.

По результатам программного и аппаратного тестирования была составлена сравнительная таблица выборок скоростных характеристик при пересылке пакетов. Каждая выборка состоит из 1000 измерений. В таблице 8 представлены усредненные, минимальные и максимальные значения, определенные по снятым характеристикам. На рисунке 32 представлен график со скоростными выборками передачи пакетов.

Таблица 8 – Скоростные характеристики передачи пакетов

№ выборки	1	2	3	4	5
Среднее значение, мс	62,2939	64,5	64,5	50,8	56
Максимальное значение, мс	115,436	140	123	130	124
Минимальное значение, мс	26,773	19	15	15	13

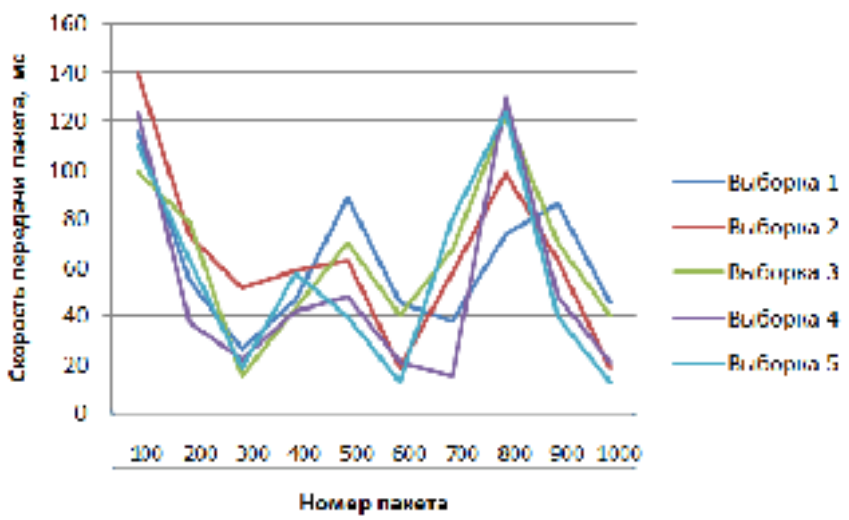


Рисунок 32 – График выборок скорости передачи пакетов

Проанализировав представленную сравнительную таблицу и график можно заметить, что время передачи пакетов в среднем велико. Это обуславливается тем, что:

1. Используется малая рабочая частота Ethernet модуля.
2. Не соответствуют рабочие частоты микроконтроллеров (ATmega328 – 16МГц, ECN28J60 – 25МГц).

Следовательно, такой протокол не выгоден при использовании в реальных устройствах. Решение следует искать при создании подобных систем на аппаратном уровне, например на основе ПЛИС. Это позволит выполнять программную реализацию оригинальных протоколов на HDL-языке и их аппаратную отработку в виде СБИС.

Тем не менее, полученные решения полностью соответствуют заданию на выпускную квалификационную работу. Сетевой контроллер предназначен для разработки и отладки нестандартных протоколов верхнего уровня в стандарте Ethernet. Состав аппаратуры и разработанного программного обеспечения позволяет решать задачи отладки, тестирования, прошивки микроконтроллера и пр.

Примененный принцип «каркасной сборки» при организации программного обеспечения позволяет выполнять его модификацию с минимальной коррекцией кода.

СПИСОК СОКРАЩЕНИЙ

АО НПЦ	– Акционерное общество Научно-производственный центр
АЦП	– Аналого-цифровой преобразователь
ВКР	– Выпускная Квалификационная Работа
МК	– Микроконтроллер
ОС	– Операционная система
ПО	– Программное обеспечение
ЛВС	– Локальная вычислительная сеть
СБИС	– Сверхбольшая интегральная схема
ШИМ	– Широтно-импульсная модуляция
CAN	– Controller Area Network
DIP	– Dual in-line Package
I2C	– Inter-Integrated Circuit
IEEE	– Institute of Electrical and Electronics Engineers
IrDA	– Infrared Data Association
LIN	– Local Interconnect Network
MAC	– Media Access Control
PCIe	– Peripheral component interconnect Express
PHDL	– Hardware description language
PHY	– Physical layer
PIC	– Programmable integrated circuit
PTP	– Point-to-Point
SOIC	– Small-Outline Integrated Circuit
SPI	– Serial Peripheral Interface
TCP/IP	– Transmission Control Protocol/Internet Protocol
UART	– Universal Asynchronous Receiver-Transmitter
USB	– Universal Serial Bus

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Капустин, В. Е. Информационно–вычислительные сети : учебное пособие / В. Е. Капустин, Дементьев. – Ульяновск : УлГТУ, 2011. – 141с.

2. Урбанович, П. П. Защита информации и надежность информационных систем : учеб. пособие / П. П. Урбанович, Д. В. Шиман, М. С. Шмаков. – Минск : БГТУ, 2012. – Ч. 1 – 42 с.

3. Элвис [Электронный ресурс] : Официальный сайт АО НПЦ "Элвис". – Москва, Зеленоград. – Режим доступа: http://multicore.ru/.

4. Миландр [Электронный ресурс] : Официальный сайт Миландр. – Москва, Зеленоград. – Режим доступа: https://www.milandr.ru/.

5. Непомнящий, О. В. Микропроцессорные системы [Электронный ресурс] : электрон. учеб. пособие / О. В. Непомнящий. – Красноярск : СФУ, 2010. Режим доступа: https://e.sfu-kras.ru/course/view.php?id=1487

6. Mirochip [Электронный ресурс] : Официальный сайт Microchip. – Санкт-Петербург. – Режим доступа: http://microchip.com.ru/PIC18F/Ethernet.html.

7. Intel [Электронный ресурс] : Официальный сайт Intel – Калифорния, США. – Режим доступа: https://www.intel.ru/content/www/ru/ru/products/network-io/ethernet/controllers.html.

8. WIZnet [Электронный ресурс] : Официальный сайт WIZnet. – Корея. – Режим доступа: http://www.wiznet.io/product-item/wiz750sr/.

9. Кондаков, Е. В. Импульсные преобразователи и стабилизаторы напряжения : учеб. метод. пособие / Е.В. Кондаков. – Ростов-на-Дону. : ЮФУ, 2014. – 41с.

10. Wikipedia [Электронный ресурс] : Proteus. – Режим доступа: https://ru.wikipedia.org/wiki/Proteus(система_автоматизированного_проектирования).

11. NI Multisim [Электронный ресурс] : – Режим доступа: http://cxem.net/software/multisim.php.

12. LabVIEW [Электронный ресурс] : Официальный сайт National Instruments. – Режим доступа: http://www.ni.com/ru-ru/shop/labview.html.

13. Atmel Studio 7 [Электронный ресурс] : Atmel Studio 7. – Режим доступа: https://www.microchip.com/avr-support/atmel-studio-7.

14. Arduino IDE [Электронный ресурс] : Официальный сайт ПО Arduino IDE. – Режим доступа: https://www.arduino.cc/en/main.

15. CodeVisionAVR [Электронный ресурс] : – Режим доступа: http://cxem.net/software/codevisionavr.php.

16. Wireshark User`s Guide [Электронный ресурс] : Документация по работе с ПО Wireshark. – Режим доступа: https://www.wireshark.org/download/docs/user-guide.pdf.

17. NET-Simulator [Электронный ресурс] : Моделирования работы сети в NET-Simulator / А. А. Коробецкая. – Режим доступа: http://kornast.ucoz.ru/Seti/seti_l.r.1-proektirovanie_seti_v_net-simulator.pdf.

18. Wikipedia [Электронный ресурс] : – Библиотека Pcap. – Режим доступа: https://ru.wikipedia.org/wiki/Pcap.

19. Антонов, А. К. Защита информации. Методы защиты информации : курс лекций / А. К. Антонов, В. М. Артюшенко. – Москва : ГОУВПО МГУС, 2005 – Ч.1. – 191 с.

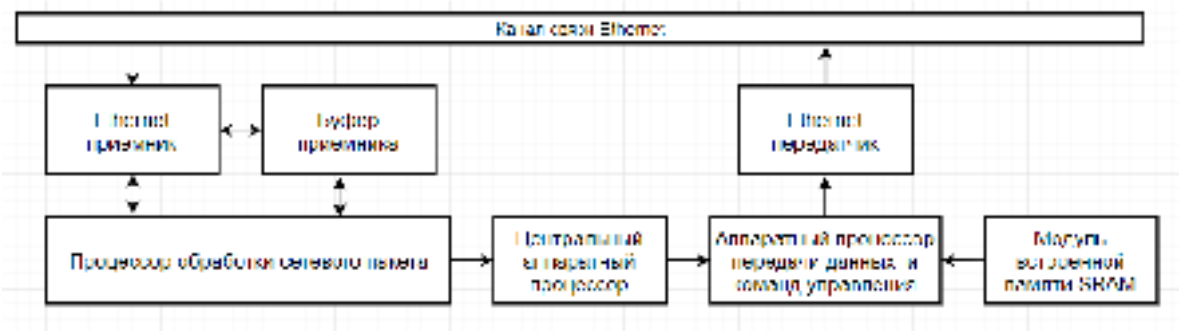
20. Datasheet of ENC28J60 : Microchip Technology Incorporated – U.S.A. : 2006 – 2012 – 102 с.

21. Datasheet of 13F-67 Series : Yuan Dean Scientific CO – LTD : 2009 – 2с.

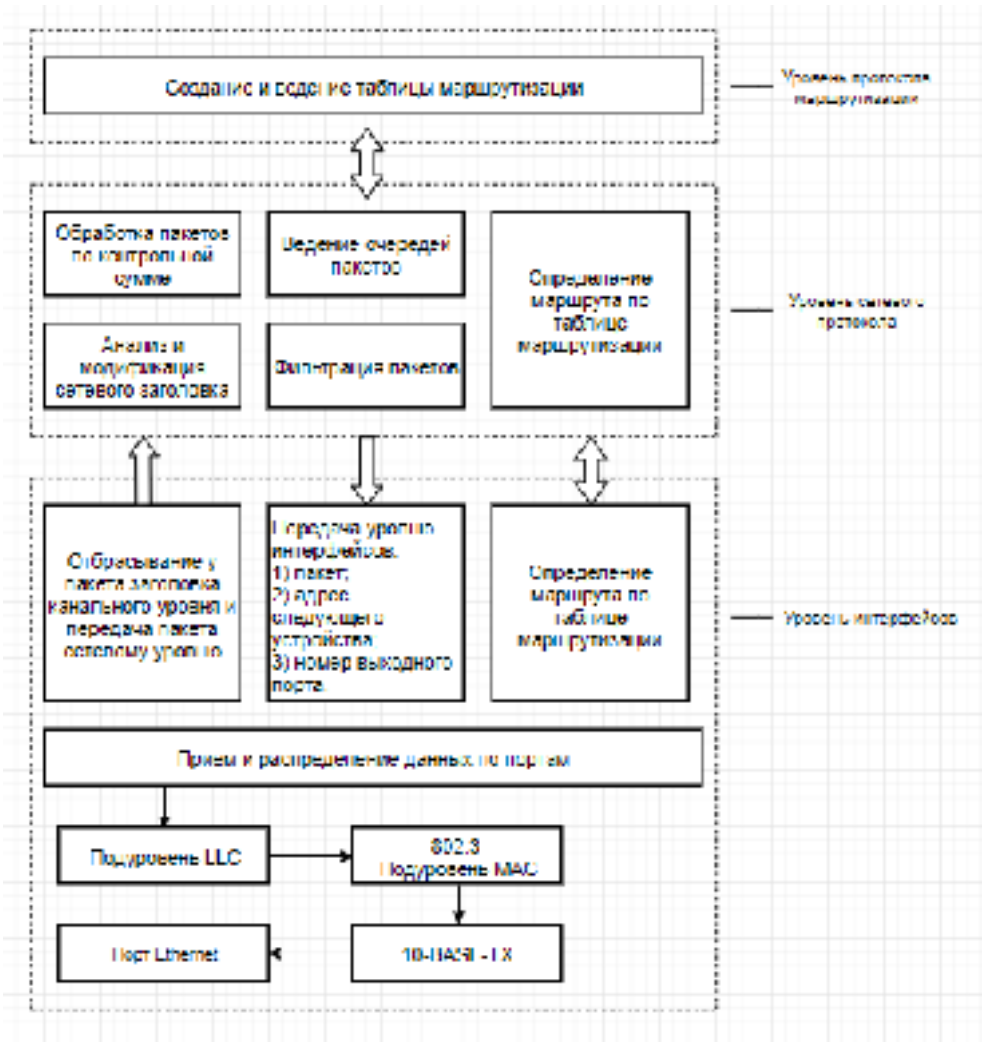
22. Бутырин, П. А. Электротехника : учебник / П. А. Бутырин, О. В. Толчеев, Ф. Н. Шакирзянов; под ред. П. А. Бутырина. – Москва. : Академия, 2006. – 268 с.

ПРИЛОЖЕНИЕ А

Функциональная схема модуля обработки пакета на аппаратном уровне

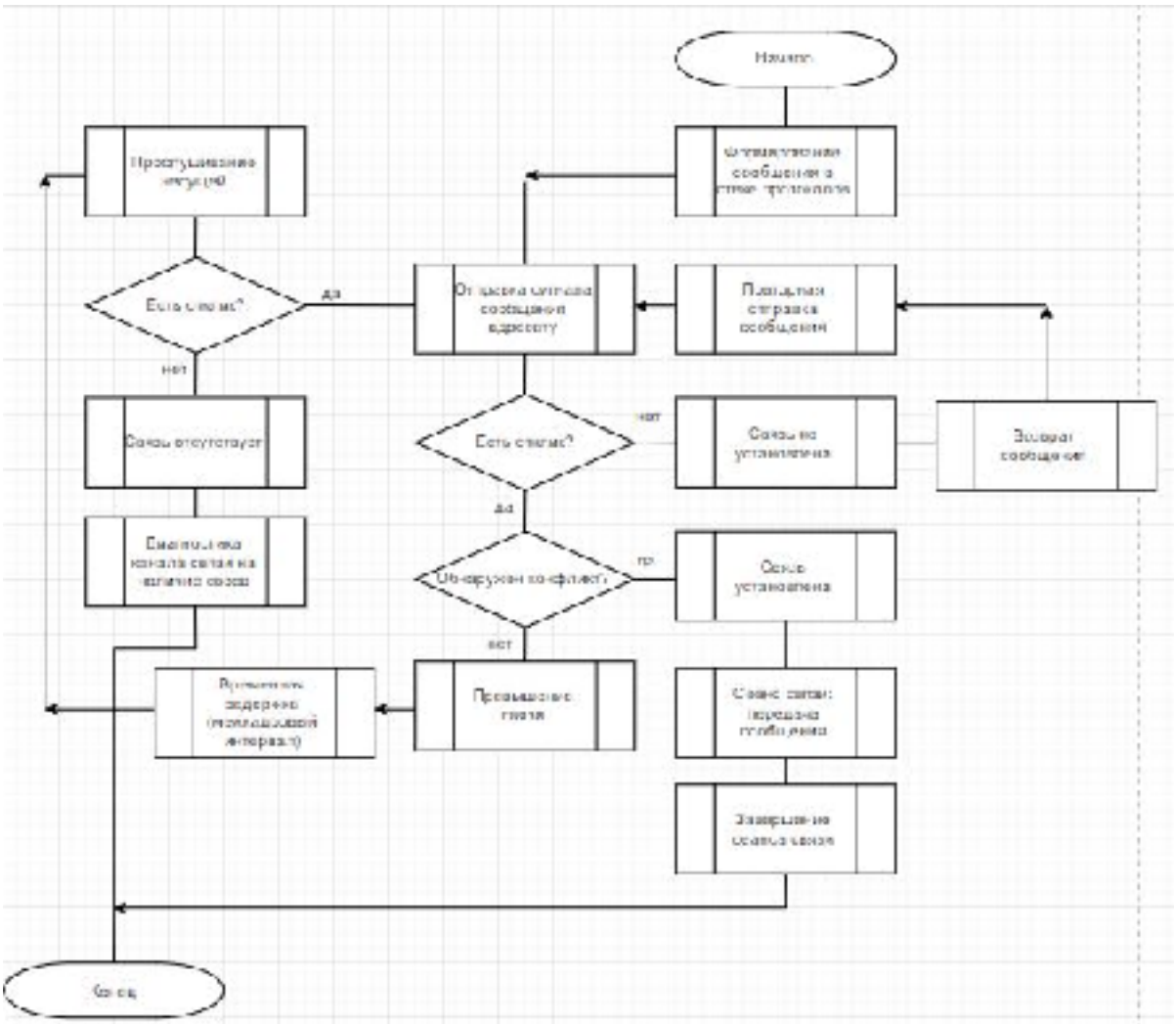


Уровни работы программного обеспечения (модель OSI)



ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А

Блок – схема алгоритма установления связи по каналу Ethernet



ПРИЛОЖЕНИЕ Б

Фрагмент листинга кода программы

test.ino

```
#include "etherShield.h"
#include "ETHER_28J60.h"

//задаем IP, MAC и порт доступа
static uint8_t mac[6] = {0x54, 0x55, 0x58, 0x10, 0x00, 0x24};
static uint8_t ip[4] = {192, 168, 0, 3};
static uint16_t port = 80;

ETHER_28J60 ethernet;

void setup()
{
  ethernet.setup(mac, ip, port);
}

void loop()
{
  if (ethernet.serviceRequest())
  {
    ethernet.print("<title>Test Page</title>");
    ethernet.print("<H1>Hello World</H1>");
    ethernet.print("<a href=start><button>Open</button</a>");
    ethernet.print("<br>");
    ethernet.print("<a href=LED> <button>LED IS ON</button</a>");
    ethernet.respond();
  }
  delay(100);
}
```

ETHER_28J60.h

```
#ifndef ETHER_28J60_h
#define ETHER_28J60_h
#include <inttypes.h>

class ETHER_28J60
{       public:
        void setup(uint8_t macAddress[], uint8_t ipAddress[], uint16_t port);
        char* serviceRequest();       //возвращает значение char*, содержит requestString
или NULL если запрос отсутствует
```

```
        void print(char* text);         //добавляем данные в ответ
        void print(int value);  //добавляем номер в ответ
        void respond();         //составляем окончательный ответ
private:
        uint16_t _port;
};
#endif
```

ETHER_28J60.cpp

```
#include "etherShield.h"
#include "ETHER_28J60.h"
#include "Arduino.h"            //под версию Arduino 1.0

//Definitions
#define BUFFER_SIZE 500
#define STR_BUFFER_SIZE 32
static uint8_t buf[BUFFER_SIZE+1];
static char strbuf[STR_BUFFER_SIZE+1];
EtherShield es = EtherShield();
uint16_t plen;

//Constructors, User API
void ETHER_28J60::setup(uint8_t macAddress[], uint8_t ipAddress[], uint16_t port)
{
        _port = port;
        es.ES_enc28j60Init(macAddress);
        es.ES_enc28j60clkout(2); //меняем задержку с 6.25МГц до 12.5МГц
        delay(10);
        es.ES_enc28j60PhyWrite(PHLCON,0x880);
        delay(500);
        es.ES_enc28j60PhyWrite(PHLCON,0x990);
        delay(500);
        es.ES_enc28j60PhyWrite(PHLCON,0x880);
        delay(500);
        es.ES_enc28j60PhyWrite(PHLCON,0x990);
        delay(500);
        es.ES_enc28j60PhyWrite(PHLCON,0x476);
        delay(100);
        es.ES_init_ip_arp_udp_tcp(macAddress, ipAddress, _port);
}

char* ETHER_28J60::serviceRequest()
{
        uint16_t dat_p;
        int8_t cmd;
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б

```
		plen = es.ES_enc28j60PacketReceive(BUFFER_SIZE, buf);
		//plen не равен нулю, когда получен допустимый пакет (без ошибки контрольной суммы)
		if(plen != 0)
		{
			//arp транслируется, если неизвестен. Происходит проверка MAC, отправляя его на unicast
			if (es.ES_eth_type_is_arp_and_my_ip(buf, plen))
			{
				es.ES_make_arp_answer_from_request(buf);
				return 0;
			}
			//проверяем, для кого пришел пакет, наш ли он
			if (es.ES_eth_type_is_ip_and_my_ip(buf, plen) == 0)
			{
				return 0;
			}
			//отправляем ответ на запрос
			f (buf[IP_PROTO_P] == IP_PROTO_ICMP_V && buf[ICMP_TYPE_P] == ICMP_TYPE_ECHOREQUEST_V)
			{
				es.ES_make_echo_reply_from_request(buf, plen);
				return 0;
			}
			//tcp port www start, сравниваем только младший байт
			if (buf[IP_PROTO_P] == IP_PROTO_TCP_V && buf[TCP_DST_PORT_H_P] == 0 && buf[TCP_DST_PORT_L_P] == _port)
			{
				 if (buf[TCP_FLAGS_P] & TCP_FLAGS_SYN_V)
				{
					//make_tcp_synack_from_syn производит отправку syn,ack
					es.ES_make_tcp_synack_from_syn(buf);
					return 0;
				}
				if (buf[TCP_FLAGS_P] & TCP_FLAGS_ACK_V)
				{
					es.ES_init_len_info(buf); //инициализируем структуру данных
					dat_p = es.ES_get_tcp_data_pointer();
					//проверяем на наличие данных (возможен просто ask)
					if (dat_p == 0)
					{
						if (buf[TCP_FLAGS_P] & TCP_FLAGS_FIN_V)
						{
							es.ES_make_tcp_ack_from_any(buf);
						}
						return 0;
					}
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б

```
                                        if (strncmp("GET ", (char *) & (buf[dat_p]), 4) != 0)
                                        {
                                                plen = es.ES_fill_tcp_data_p(buf, 0, PSTR("HTTP/1.0
200 OK\r\nContent-Type: text/html\r\n\r\n<h1>200 OK</h1>"));
                                                plen = es.ES_fill_tcp_data_p(buf, plen,
PSTR("<h1>A</h1>"));
                                                respond();
                                        }
                                        //was "/" and 2
                                        if (strncmp("/", (char *) & (buf[dat_p + 4]), 1) == 0)
                                        {
                                                //копируем содержимое запроса, прежде чем
перезаписать его ответом
                                                int i = 0;
                                                while (buf[dat_p + 5 + i] != ' ' && i <
STR_BUFFER_SIZE)
                                                {
                                                        strbuf[i] = buf[dat_p + 5 + i];
                                                        i++;
                                                }
                                                strbuf[i] = '\0';
                                                plen = es.ES_fill_tcp_data_p(buf, 0, PSTR("HTTP/1.0
200 OK\r\nContent-Type: text/html\r\n\r\n"));
                                                return (char*) strbuf;
                                        }
                                }
                        }
                }
        }

        //добавляем данные в ответ
        void ETHER_28J60::print(char* text)
        {
                int j = 0;
                text = Encode(text, "coding");
                while(text[j])
                {
                        buf[TCP_CHECKSUM_L_P + 3 + plen] = text[j++];
                        plen++;
                }
        }

        //добавляем номер в ответ
        void ETHER_28J60::print(int number)
        {
                char tempString[9];
                itoa(number, tempString, 10);
```

```
        print(tempString);
}
//составляем окончательный ответ
void ETHER_28J60::respond()
{
        es.ES_make_tcp_ack_from_any(buf); //отправляем ack для запроса http GET
        es.ES_make_tcp_ack_with_data(buf,plen); //отправляем данные
}
```

Листинг разработанного программного обеспечения в полном объеме приведен на CD-диске.